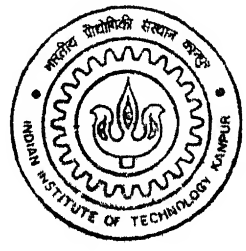
A CONSTRAINT-SATISFACTION HEURISTIC FOR SCHEDULING TELEMETRY, TRACKING AND COMMANDING OPERATIONS OF IRS SATELLITES

by
Garima Shahi



DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING
Indian Institute of Technology Kanpur
April, 2001

A CONSTRAINT-SATISFACTION HEURISTIC FOR SCHEDULING TELEMETRY, TRACKING AND COMMANDING OPERATIONS OF IRS SATELLITES

A Thesis Submitted in

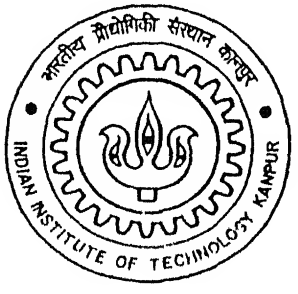
Partial Fulfillment of the Requirements

for the degree of

Master of Technology

by

Garima Shahi

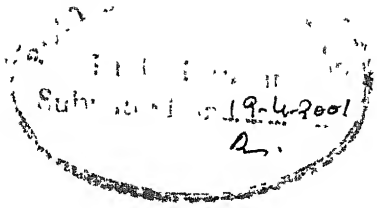


to the

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

April, 2001



CERTIFICATE

It is certified that the work contained in this thesis entitled ***"A Constraint-Satisfaction Heuristic for Scheduling Telemetry, Tracking and Commanding Operations Of IRS Satellites"*** has been carried out by Ms Garima Shahi (Roll No 9911405) under my supervision and this work has not been submitted elsewhere for a degree

T P Bagchi
Professor,
Dept. of Industrial and Management Engineering,
Indian Institute of Technology,
Kanpur – 208016

18 April,2001

ACKNOWLEDGEMENTS

My special and sincere thanks to Dr. T.P. Bagchi for his invaluable guidance, inspiration and encouragement throughout the course of this work. I owe him a strong debt of gratitude for his unflagging support, for the confidence he showed in my capabilities and for his cheerful readiness to help me in all possible ways throughout the days of my hard work

I extend my thankfulness to staff of ISTRAC – Mr. P. Soma, Mr. Venkateshwar Lu, Mrs. SanthaLakshmi, Mr J.D. Rao, Mrs. Padma Rani and Mr. Prakash Rao for explaining the problem to us and for the cooperation they showed to us right from the beginning. Our work has benefited much from the stimulating discussions with them.

Thanks is in store for Dr. Mittal and Dr. Swami who taught me the fundamentals of Operations Research and Scheduling. I also thank all faculty members for kindling my knowledge in various courses of Industrial Engineering.

I express my gratitude to my friends - Ashish, Yamini, Pooja, Reena and Kriti and Vishwesh for giving me their pleasant company and for cheering me during my tough times. Memories of the moments spent with them will remain fresh in my mind Thanks to Meenu, Makarand, Sagar and Ramakrishna for showing their promptness to help me during my tough times.

To my parents I owe more than what I can mention ... especially for the care they took for me even when I was miles away from them and for tracking me to see the silver lining in every dark cloud. They have been a source of inspiration in my life and career, thus far.

Last but not least, I am grateful to the almighty God who gave me courage to do the job efficiently.

26 April, 2001

Garima Shahi

ABSTRACT

Satellite network operations can be viewed as a network of satellites and ground stations that require frequent contact links between them for satellite tracking, maintenance and data dissemination Contact windows between a satellite and a ground station are temporal and location dependent. Scheduling the operations for assignment of network resources to meet concurrent contacts (via transmitting and receiving antenna at ground stations) and other operational requirements without violating temporal, compatibility and capacity constraints, is a complex problem, it requires an extensive knowledge of the system and subsystem operations, operational constraints, ground station resources, and satellite design and configuration. In addition, constraints, priorities, mission requirements and resource availability are dynamic and vary between satellites Further, satellite- scheduling problems require the introduction of additional variables (satellites, chains, operations), this adds to complexity of the problem. The traditional techniques - linear programming (LP), quadratic programming (QP) and mixed integer programming (MIP) do not fit well to solve such complex problem; no text book models exist that are of direct utility to solve this problem. In this thesis, we propose a constraint based heuristic for scheduling contacts between LEO (Low Earth Orbiting) satellites and ground stations to perform maximum number of operations on these satellites from telemetry, commanding and tracking stations of the ground network, providing minimum required coverage at each of the scheduled windows and respecting the various hard constraints - temporal constraints, resource availability constraints, physical constraint, etc. This approach is more intuitive and flexible as compared to traditional techniques of optimisation and fits well for applications in which users want good, feasible answers rather than proven optimality.

CONTENTS

Chapter 1: LEO Satellites Scheduling: A Challenge

Scheduling is a ubiquitous activity in everyday life. A major field of research in scheduling is the problem abstracted as "machine scheduling", the solution of which has applications in manufacturing, logistics, computer architecture design, healthcare administration, air transport management, etc. Some specific applications of machine scheduling are short-term production planning, workforce scheduling and time tabling (Bagchi, 2000). Scheduling affects both the private and public sectors (Zweben and Fox, 1994) In the industrial sector scheduling is key to manufacturing, distribution, transportation, engineering maintenance, entertainment production, and construction. In government, scheduling is also mission critical. All of NASA'S spacecraft operations, both in space and ground, are meticulously scheduled In a manufacturing system, orders have to be released and have to be translated into jobs with associated due dates; a number of tasks are involved in this. Detailed scheduling of these tasks is necessary to maintain efficiency of production system – whether job shop, flow shop or open shop. Scheduling problems are known to be NP-hard and proven to be a difficult task for human planners and schedulers. In this thesis, we propose and implement a heuristic to optimally schedule the operations support of Indian remote sensing satellites on telemetry, tracking and commanding ground stations.

1.1 What is Remote Sensing?

According to the Manual of Remote Sensing, *remote sensing* (web reference 1) is the "measurement of or acquisition of information of some property of an object or phenomenon, by a recording device that is not in physical contact with the object or phenomenon under study." Nearly all remotely sensed data is collected by sensors on one of two platforms: satellites or aircraft.

Satellite-borne sensors have become the most common sources of remotely sensed data in geographic research These sensors allow for a large area of the Earth's surface to be captured in one pass of the satellite. They also allow for multiple passes of the same area over time. This provides the ability to study geographic phenomena at

broader scales than are possible from the ground and to investigate changes in these phenomena over time

There are two types of satellite sensors, passive and active. Passive sensors intercept reflected electromagnetic radiation off of the surface of the planet and convert it to a digital form. These multispectral sensors can separate the incoming radiation into a number of spectral bands (see Spectral Resolution). Active sensors send out energy, usually in the form of microwaves, and then intercept the energy that is reflected off the the surface

Some of the most commonly used data from passive sensors comes from the Landsat, SPOT, and AVHRR sensors (web reference 1). An example of an active sensor is RADARSAT, a Canadian radar sensor. One of the newest sensors to recently go into orbit is Space Imaging's IKONOS sensor, which is has a 4m spatial resolution multispectral sensor and a 1m panchromatic sensor. IKONOS is an example of the new wave of high-spatial resolution commercial satellites that are currently being developed around the world

There are three main benefits provided by remotely sensed data:

1. The ability to view a large area of the Earth's surface at one point in time, and
2. The ability to update research with new data multiple times per year
3. The ability to obtain information for areas of the earth that would otherwise be very difficult to reach, physically and/or financially

Thus, remote sensing is a great way to study both spatial and temporal phenomena.

1.2 Satellites and their Applications:

Satellite is an object, which revolves around another object. For example, moon is a satellite of the earth and the earth is a satellite of the Sun Man-made or artificial satellites are those which orbit the earth in their fixed orbits. The shape of this orbit may be elliptical or close to circular. India is having its two series of satellites: -

- Indian National Satellite (INSAT) Series
- Indian Remote Sensing (IRS) Series

Satellites of INSAT series are geo-stationary (moving at a height of around 36000 Km. and therefore having time-period of 24 hrs). These satellites are used for communication, broadcasting and meteorological purposes.

Satellites of IRS series are low earth orbit (LEO) satellites moving at a height of around 700 to 900 km (time period is around 100 minutes) These satellites form a critical infrastructure for India's natural resource management, are used for a number of remote sensing applications in different areas such as, agriculture and crops, forest and bio-resources, flood control, ocean study, rural development, urban planning, space research support, etc India is a leading developing country and science of remote sensing is of particular relevance for its social and economic development, as described by Kasturirangan (2000).

Satellites of IRS series are. -

- 1A, 1B, 1C, 1D, P2, P3, P4, and SROSS .

These satellites are fabricated at ISRO Satellite Centre (ISAC), Bangalore and launch vehicle is fabricated at Tiruvananthapuram Launching of satellite into required orbit is done from Shriharikota. The main purpose of IRS series of satellites is nation's resource monitoring and its management. Besides this, satellites are generating revenues by taking images of the locations on the globe as requested by customers – Indian and foreign. Today a minute's worth of remote sensing can bring upwards of $3500 to the service provider in the open market.

1.3 Ground Trace of Satellite:

On projecting the satellite on to earth, the point obtained is called *sub-satellite point* and locus of sub-satellite points is called *ground trace* Although the path of a satellite is a fixed orbit, by virtue of spin motion of earth, its ground trace is spiral shaped (Figure 1.1). Also the spiral traversed each day is not exactly the same as that of previous day The spiral path covered on a particular day is repeated after a fixed period of time, which may be different for different satellites. This period of time is called *repeat cycle* of the satellite (IRS-P4 → 2 days, IRS-1C → 24 days, etc.) In other words, repeat cycle of a satellite is a set of different paths the satellite can follow.

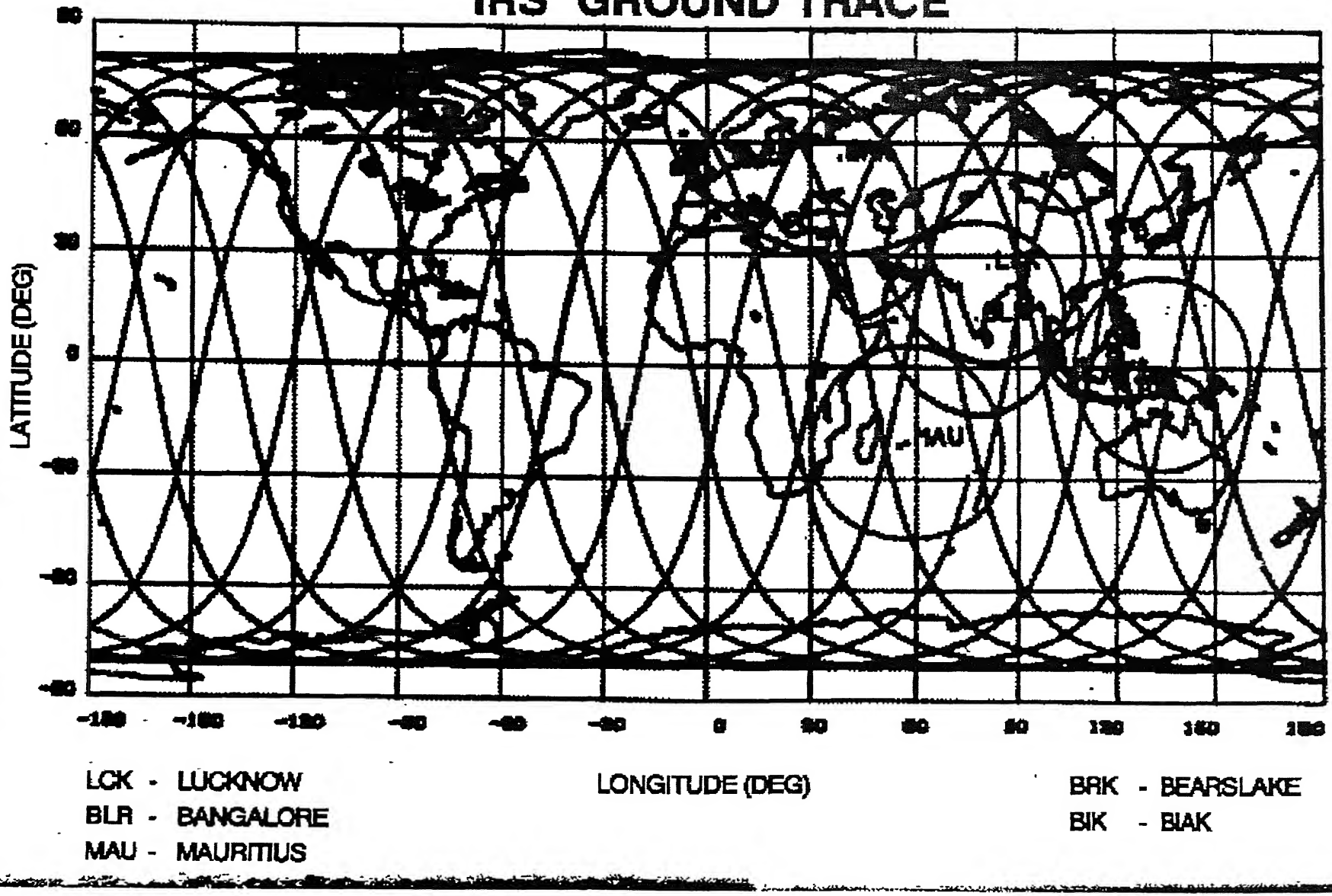

IRS GROUND TRACE
LATITUDE (DEG)
LONGITUDE (DEG)
MAU
LCK - LUCKNOW
BLR - BANGALORE
MAU - MAURITIUS
BRK - BEARSLAKE
BIK - BIAK

FIGURE 1·1

As explained above, a satellite covers different points of globe at different points of its motion. It is very important to monitor the "health parameters" of various sub-systems of the satellite continuously and also keep track of its orbit. This is achieved by means of special configurations set up at geographically distributed locations and which can establish radio-link with the satellite and thus can communicate (send and receive signals) with it. These configurations are called *ground-stations*. These ground-stations are exclusively of two types: -

- Telemetry, Tracking and Commanding (TTC) stations, and
- Payload (PL) stations.

A set of two or more stations is called *ground network*. Ground stations of ISRO (Indian Space Research Organization) are distributed all around the globe. A satellite is said to be visible for a ground station if it is above the local horizon of the ground station. The time interval for which a satellite is visible on a ground station is called *visibility window* of the satellite. Each window is associated with two delimiters: -

- AOS (acquisition of signal)
- LOS (loss of signal)

AOS is the time instant from which exchange of signals between a satellite and a ground station can be started. *LOS* is the time instant beyond which exchange of signals between a satellite and a ground station is not possible. In other words, AOS is the instant of time when a satellite 'rises' and LOS is the instant of time when it 'sets' with respect to a ground station. 'Maximum' length of a visibility window (duration between AOS and LOS) of a LEO satellite on any ground station is around 16.3 minutes.

1.4 Operations on Satellites:

Different types of operations require to be performed on the satellites for various purposes-continuous monitoring of its health, maintaining its position in fixed orbit, commanding the timers, resetting its various devices, etc. These operations on LEO satellites are mainly classified as:

- **TTC operations:** These operations are required for the monitoring of health of the satellite subsystems

- **Payload operations:** Payload operations are used in imaging the ground from LEO satellites.
- **Command Operations:** These are very small duration operations performed to send signals to the space-craft for resetting its various devices, switching ON/OFF the cameras, commanding the timer, etc.

Operations of a satellite are performed on the visibility windows of the satellite over various ground stations Operations of first category require to be performed on visibility windows over TTC stations respecting various constraints specific to the satellite-scheduling domain. *TTC network* (which is a subset of ground network, comprising of TTC stations) provides dedicated services for these operations throughout the mission life of the satellite. *Payload stations* receive data from payload devices mounted on the satellites according to the planned schedule.

1.5 Scheduling Satellite Supports and Complexities Involved:

Scheduling concerns the allocation of limited resources to tasks over time (Pinedo 1995) It often involves resources that are not continuously available, get depleted by use or are subject to failure or deterioration. Scheduling problems are generally integer programming problems and would therefore involve standard integer programming techniques such as branch-and-bound and other heuristics. Satellite scheduling like all scheduling, is the problem of mapping tasks (observation, communication, downlink, control maneuvers, etc.) to resources (satellites, ground stations, etc.) but these are highly complex and notoriously difficult to solve. The difference of satellite-scheduling problems from traditional scheduling problems lies in: -

- Foremost is the fact that exhaustive enumeration of the possible schedules is usually impractical.
- Further complications arise because of the different types of constraints that plague a real-world application (Fox 1994). Temporal constraints, geometric constraints, maneuvering constraints, etc. make the problem extremely complex
- Choosing an objective function can be difficult One may choose to· -

i. maximize the number of targets scheduled, or

ii. minimize the 'idle' time of the satellite, or

iii. schedule maximum number of tasks, or

iv maximize the total time of support, or

v minimize the total number of supports required, or

vi maximize quality of collected data, or

vii maximize the scheduling flexibility provided to the tracking station, or

viii. maximize the total 'score' benefit function

- Satellite-scheduling problems oftentimes involve periodic tasks, variable length tasks, and tasks that can be preempted.
- An obvious difference between satellite scheduling and traditional scheduling problems is the fact that some of the resources (i e. satellites) are orbiting the earth. This places an additional set of constraints on when a task can be executed.

Satellite scheduling belongs to a class of problems whose complexity increases exponentially with the number of satellites as well as the umber of operations These are generally thought to belong to the class of problems which are NP-complete, that is, no algorithm exists which can solve this class of problems in polynomial time. Also, the problem cannot be mapped to standard machine scheduling paradigms or to mathematical methods such as integer programming Defining an accurate objective function is difficult, although not impossible. The "weighting factors" that are often introduced, create dubious measures as they attempt the proverbial combination of "apples" and "oranges" Due to the complexity of satellite-scheduling problems, satellite support scheduling is now commonly carried out according to an assortment of meta-heuristic and artificial intelligence (AI) procedures that consider mission requirements and constraints specific to the problem of satellite - scheduling.

1.6 Literature Review:

- Resource allocation problems involve making many decisions, including which set of resources are to be assigned to each demand (ILOG, 1998). Each decision has a limited number of possible assignments and must satisfy the operational constraints. These attributes –multiple decisions, limited outcomes and constraint satisfaction – define a class of problems called combinatorial optimization problems (COP). Scheduling problems fall into this class. For a COP, a series of decisions must be made while at the same time satisfying a set of constraints and optimizing a cost function The solution should optimize the cost functions and handle as many preferences and constraints as possible.

There are many approaches to optimization, these fall into two broad categories: global and local optimization approaches Global optimization will always find the same solution, regardless of the starting point, but usually requires more computational power. Sometimes finding a good solution, a local optimum, quickly is more desirable than finding the best solution.

To represent decisions that need to be made in an optimization problem, decision variables are used For example, a cereal company may want to know what proportion of their new cereal should be made of shredded wheat. This decision is represented by a real variable in the range *[0,I]* As another example, a machine scheduler may want to know which component to process first among three components (A, B and C). The decision is represented by a discrete variable with possible values in *{A, B, C}*

There are many different types of variable.

- Real variables have a lower and upper bound (l and u respectively) and may take any value in the range *[l, u]*.
- Integer variables have a lower and upper bound (l and u respectively) and may take any integer value in the range *[l, u]*.
- Logical variables may be either true or false. These variables may be represented by an integer variable with a lower bound of 0 (for false) and an upper bound of 1 (for true).

- Choice variables have a set of possible values (e g , the bar codes of different products) and must take one of these values These variables are often represented by an integer variable with a lower bound of 1 and an upper bound of the number of possible values The integers in this range correspond to different choices
- Set variables have different sets as possible values (e.g., the set of products to store in a container) and must be one of these sets These variables may be represented as choice variables, but their interactions with other variables are often more complex

The best approach to solving an optimization problem depends on the structure of the problem – type of decision variables (real/integer/logical/choice...), constraints (linear/nonlinear), and objective function (linear/nonlinear) the problem includes Linear functions involve a linear combination of decision variables whereas nonlinear functions involve any function of decision variables, covering a wider range of interactions and tradeoffs that may occur in real- world situations.

Figures 1 2-(a) and 1 2- (b) on next page, summarize the best match between the application types (problem structure) and optimization techniques.

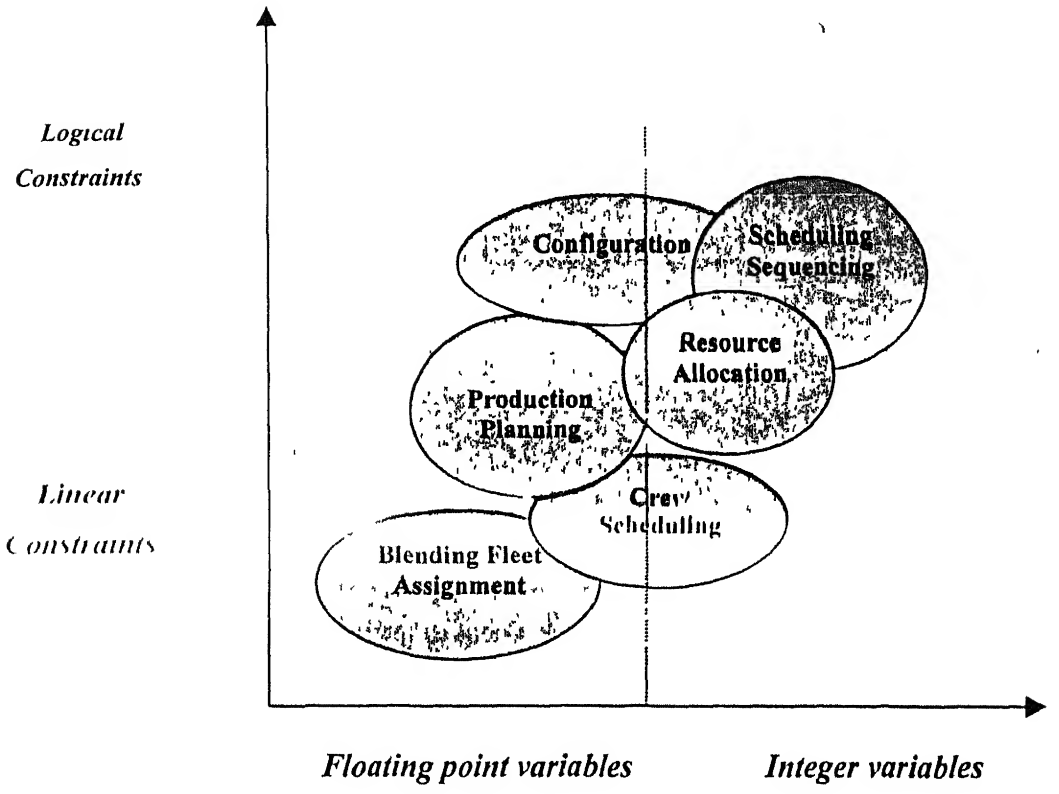
Orbiting once every 100 minutes or so at around 800 km height the LEO (Low Earth Orbiting) satellites pass through various stations of the ground network (positioned all around the globe) and are visible for a few minutes over any station. From these ground stations, the satellites require to be routinely executed. Number of supports for each of the operations on all the satellites is determined by the planner analysts of the mission Scheduling these satellite supports is difficult, although not impossible, because of the nature of constraints and objective function. Complete knowledge of the system and enough expertise is required to plan the schedule. Process is much time consuming.

- A 'number of machines in parallel ' is a setting that is important from both the theoretical and practical points of view. From the theoretical viewpoint, it is a generalization of the single machine. From practical point of view it is important because 'occurrence of resources in parallel' is common in real world Satellite scheduling problem resembles parallel machine scheduling (Agnese et al, 1995)

Application types and optimization techniques

(ILOG optimization suite white paper)

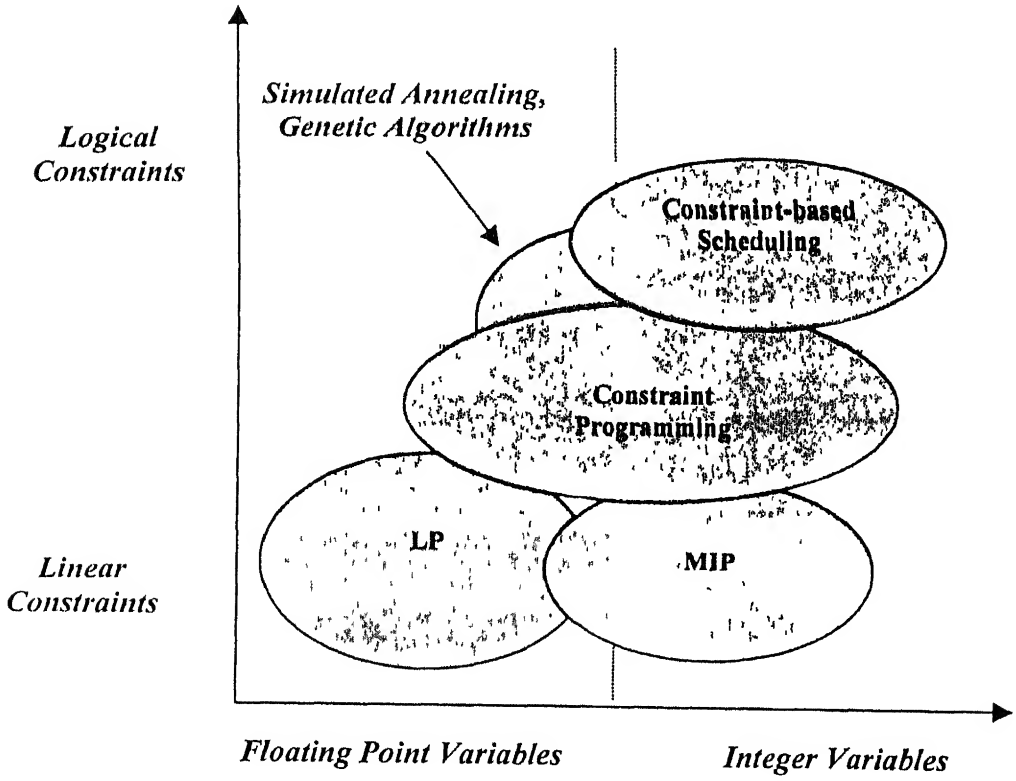
Figure 1.2-(a)



Application types and optimization techniques

(ILOG optimization suite white paper)

Figure 1.2-(b)



This problem is NP-complete (Agnese and Brousse, 1998).

- Agnese and Browsse (1998) describe various combinatorial optimization techniques for the resolution of satellite-scheduling problem. Exact methods like Depth First Branch and Bound, Russian Dolls (hybridization of Dynamic Programming and Branch and Bound methods), Linear Programming, have the advantage of providing optimal solution at the price of very large computation time. Approximate methods, like random greedy search (iterative or random) have the advantage to provide good quality solutions within limited time.
- A system for NASA's terriers satellites based on dispatching rules was implemented in 1992 without the explicit consideration of optimality (Groleau, 1992).
- Satellite scheduling problem was modeled as a linear programming problem in 0-1 numbers and the model was first implemented using XPRESS. On a typical problem with a set of 870 initial windows, the model led to more than 380000 constraints. The only preprocessing took more than 1 hour on a SUN SS30 workstation and was very long to solve
- A prototype expert system for scheduling supports of satellites in Global Positioning System has been developed by Kennedy et al (1988). It uses scheduling rules and heuristic methods to solve the problem. The two main objectives that the scheduler aims to achieve are to minimize the total number of supports required for each satellite and to provide the scheduling flexibility provided to the tracking station schedulers.
- AI (Artificial Intelligence) methods have been used to represent constraints and for searching good schedules to determine near-optimal long term scheduling of Hubble Telescope Observations. The methods used heuristic logic and hill-climbing schedule repair schemes to determine a hierarchical ordering of activities to schedule the most constrained ones first.
- Analysis of contact times for a number of spacecraft, so that resources can be properly scheduled and facilities are utilized in a most efficient and cost-effective way, is a typical task for ground station operators. SatTrack, a commercial system, can easily provide simultaneous schedules of contact times for up to 100 spacecraft and up to 10 ground stations. A number of constraints can be applied to the contact

schedule fixed elevation, station mask, maximum range, minimum link margin, sensor and antenna angles, minimum pass duration, and minimum solar elongation of the spacecraft. In addition, pre-pass and post- pass periods can be included. The scheduling tool also provides an analysis of conflicts and a statistical summary of contact intervals

A sample schedule produced by SatTrack is shown below for five spacecraft (FAST, TRACE, NOAA 14, TOMS-EP, and SWAS) and two ground stations (Wallops Island and Berkley)· -

Contact schedule sorted by acquisition time for station Wallops Island

TRACE	25280	2000 / 026	00.20·51	– 2000 / 026	00:30:39	UTC	9 8 min
SWAS	25560	2000 / 026	00.45·11	- 2000 / 026	00:49·16	UTC	4 1 min
TOMS-EP	23940	2000 / 026	02.44.02	– 2000 / 026	02 51 06	UTC	7.1 min
FAST	24285	2000 / 026	03:25:48	– 2000 / 026	03:34:40	UTC	8.9 min
TOMS-EP	23940	2000 / 026	04.19.49	2000 / 026	04.34.02	UTC	14.2 min
FAST	24285	2000 / 026	05 36 43	– 2000 / 026	05 51 17	UTC	14 6min
TOMS-EP	23940	2000 / 026	05 59 04	– 2000 / 026	06 10 46	UTC	11 7 min

Contact schedule sorted by acquisition time for station

NOAA 14	23455	2000 / 026	00 00.00	– 2000 / 026	00 03.50	UTC	11 7 min
TRACE	25280	2000 / 026	00.24 17	– 2000 / 026	00 31 01	UTC	11.7 min
SWAS	25560	2000 / 026	00 37.41	– 2000 / 026	00 50·17	UTC	11 7 min
NOAA 14	23455	2000 / 026	01 34 15	- 2000 / 026	01 41 21	UTC	11 7 min
TRACE	25280	2000 / 026	01.57·55	2000 / 026	02·10.36	UTC	12.7 min
SWAS	25560	2000 / 026	02:17 15	– 2000 / 026	02·30:30	UTC	13 2 min
TRACE	25280	2000 / 026	03 35:01	– 2000 / 026	03 44.13	UTC	9.2 min

. . . .

Within minutes, mission planners and ground station operators have the information on their hands needed to decide how and when resources are to be assigned and how scheduling conflicts with multiple vehicles can be resolved.

- A commercial system, ILOG Telecommunications, uses integer programming rather than fractional coverage (true optimality) basis to allocate the tasks in satellite-scheduling problem It satisfies all critical constraints (web reference 2).
- The ACE Visual Scheduler (web reference 3) is one product in Braxton Technologies' line of ACE Premier satellite ground system products. The Visual Scheduler :
  - Generates and displays vehicle visibility information based on Two Line Element sets and ground terminal locations
  - Can load and process multiple file formats (e.g. PAP, DEFT, General List, TLE import files...)
  - Identifies and highlights scheduling conflicts based on user supplied data (support time, necessary equipment, turnaround, etc.)
  - Facilitates de-confliction activities
  - Generates DEFT format files for export to other systems
  - Uses a graphical drag and drop interface
  - Is a Windows NT based system, for high performance and low cost.

The ACE Visual Scheduler was originally designed to support the classified users of the U S. Air Force's Satellite Control Network (AFSCN). The AFSCN is composed of 22 ground terminals at 10 separate geographic locations around the world, and 2 operations Centres. At each of the operations centers there are multiple Mission Control Complexes (MCCs), each with their own families of satellites to support. The AFSCN currently supports 122 vehicles on orbit These satellites are predominantly in low-earth (LEO), highly elliptical (HEO), and geostationary (GEO) orbits. The MCCs submit requests to a centralized scheduling system, asking for certain ground terminal resources at specific times, in support of both health and safety (H&S) contacts and mission specific contacts).There are typically 300-350 satellite contacts performed each day, but occasionally as many as 500 supports are performed in a single day. Contact durations, specific equipment types required and ground terminal location requirements vary from hour to hour and week to week. The original scheduling activity was lengthy,

cumbersome and time intensive due to the quantity of individual users making requests, the many conflicting or overlapping requests for the same resources, the different priorities associated with the vehicles supported and the changing visibilities from ground stations to satellites The scheduling activity is occasionally perturbed even further by launch demands, which require numerous resources at very specific times, and no variance or substitution is possible. The classified users have the most complex schedule needs and wanted to generate their own internally de-conflicted resource request set in order to facilitate the overall scheduling process Where the previous system typically took eight hours to come up with a first take at a complete schedule, Braxton Technologies developed a graphically based scheduling tool that allows them to schedule 48 hours worth of de-conflicted operations in 20 minutes. The generation of the visibilities from all the AFSCN ground antennas to all of the supported on-orbit vehicles, takes less than 30 seconds

The Visual Scheduler allows users with multiple on-orbit vehicles and single or multiple ground terminals t rapidly develop a schedule for any lengthperiod (one day, 48 hours, a week, etc) that maps resource needs to time slots, using an optimization algorithm the allows the user to account for.

- specific vehicle priorities
- any necessary pre-pass times
- minimum acquisition angle above horizon
- ground terminal obscura (which can be determined automatically based on user-provided geodetics)
- other user-specified criteria

• A method *using GA* (Genetic Algorithm) has been proposed (Kumar,2000) to resolve satellite visibility clashes. GA does not guarantee optimality, but it very effectively finds sufficiently good global solutions (Goldberg, 1989; Bagchi, 1999).
Figure 1.3-(a) -displays the general picture of a single occasion of visibility clash. The problem of resolving a clash has been mathematically formulated as follows· -

V = Difference of first clashing AOS and last LOS.

I = Total number of satellite visibilities clashing in a pass over a station, ($i = 0 \ldots I$)

P = The Value Function ($f(x_1, x_2, x_3, \ldots x_n)$) to be maximized

a_i = Start of visibility (AOS) of satellite i.

b_i = End of visibility (LOS) of satellite i.

s_i = Start of support of satellite i

e_i = End of support of satellite i.

x_i ($= s_i - e_i$) = Support given to satellite i when it passes over a station.

min = Minimum duration of support once support begins

max = Maximum duration of support required

r = Station reconfiguration time (added to end of the of previous support period)

C_i = Value or profit contributed to P *per unit time* when satellite i is supported

t_{i+1} is a binary variable that indicates whether satellite i is supported or not Thus if $x_i = 0$, $t_{i+1} = 0$ and if $x_i > 0$, $t_{i+1} = 1$ for $i = 1, 2, 3, \ldots, I$; $t_1 = 0$.

Maximize $P = f(x_1, x_2, x_3, \ldots x_n)$

In general, $f(x_1, x_2, x_3, \ldots x_n)$ may be nonlinear, discontinuous, and have multiple peaks. In the special case when $f(x_1, x_2, x_3, \ldots x_n)$ is linear,

$$P = \sum_{i=1}^{I} C_i x_i = \text{total value generated} \qquad (1.1)$$

$$V = \sum_{i=1}^{I} x_i + (\sum_{i=1}^{I} t_i) * r$$

= total visibility at the ground station (1.2)

Subject to:

(i) Start of support (s_i) of satellite 'i' must be at AOS_i or later and it should be equal to or less then LOS_i.

$$b_i \geq s_i \geq a_i \tag{1.3}$$

(ii) End of support (e_i) of satellite i must be at LOS_i and it should be equal to or greater than AOS_i

$$a_i \leq e_i \leq b_i \tag{1.4}$$

(iii) Station Reconfiguration allowance

$$s_i \geq \{ \underset{k\ (i\ 1)\ 1}{Maximum}\ (e_k \times t_{k\,i}) \} + r \tag{1.5}$$

(iv) Duration of support of satellite i

$$x_i = e_i - s_i \tag{1.6}$$

(v) Constraint for minimum time of support This may be either 0 or greater than the quantity *min*. Therefore,

$$x_i = 0 \text{ or } x_i \geq min \tag{1.7}$$

(vi) Maximum time of support should be less than the quantity *max*, as specified by the decision maker.

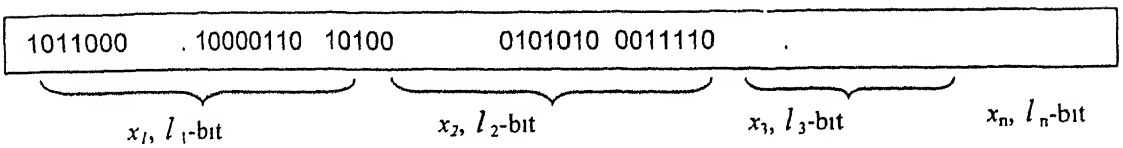
$$x_i < max \tag{1.8}$$

(vii) Nonnegativity constraint

$$s_i, e_i \geq 0, \quad x_i \geq 0\,; \tag{1.9}$$

Constraint (v) makes the problem nonlinear, even if the objective function (1.1) is linear. The search space is not convex. The profit or value function P (= $f(x_1,x_2,x_3,\ldots,x_I)$) may be a nonlinear and involve multiple local optima. Thus the feasible values of the decision variables $\{x_i\}$ are discontinuous This is a major departure from standard LP type formulations that require convexity of the feasible space. Lastly, parameters *min*, *max* and r, may be satellite- or station-specific GA casts the solution in some innovative way in a *chromosome-like* structure in order that "genetic operations" may then be performed on it A single chromosome for

satellite support contains all data for the full duration of support for each satellite involved in a clashed set of visibilities. In general, for n visibilities clashing, the chromosome itself has the following appearance (note the difference in the bit lengths $\{l_i\}$ of the different decision variables):



GA facilitates global search by creatively generating new solutions as the search progresses. In GA, "crossover" creates "progeny" by exchanging information (gene-holding *segments* of the chromosome) among "selected" "parent" strings resident in the mating pool (Goldbert, 1989). "Mutation" facilitates local search. The preserve-the-best and roulette wheel selection strategies were employed in the numerical example below.

Numerical Example:

This example assumes that four satellites are simultaneously passing over a ground station supporting them, causing their visibilities to clash. The maximum support required is uniformly 983 seconds while the minimum support required once support is decided to be given is 263 seconds. Visible time windows are as follows:

$a_1 = 616520$, $b_1 = 617115$; $a_2 = 617244$, $b_2 = 618318$;

$a_3 = 617712$, $b_3 = 618650$; $a_4 = 618027$, $b_4 = 619196$

The above data (616520, 617115, etc.) are time marks (in seconds) from a reference point. The profit function $f(x_1,x_2,x_3, ..,x_l)$—here a linear function involving profit rates C_1, C_2, C_3 and C_4 (the respective profits generated per unit time of support)—is $\sum_{i=1}^{l} C_i x_i$

To solve this problem a bit length of 10 was used for all four decision variables x_1, x_2, x_3 and x_4 (Figure 1.3-b) GA parameters p_c (the probability of crossover) and p_m (the probability of mutation) were set by running pilot GAs in a design-of-experiments framework (Bagchi, 1999) Population size was 20. Table 1.1 displays the support times determined for four different problem scenarios each with its own $\{C_i\}$ profile. A reconfiguration time (r) of 600 seconds was assumed

GA solved each scenario in Table 1.1 in about 1 second by a C++ program running on a Pentium III 500 MHz system Figure 1.3-(c) graphically displays the visibility and support periods for the profit rates $\{C_i\}$ of Case 1.It can be observed from the figure support does shift towards the satellite contributing at the maximum rate to total profit while some satellites get no support at all

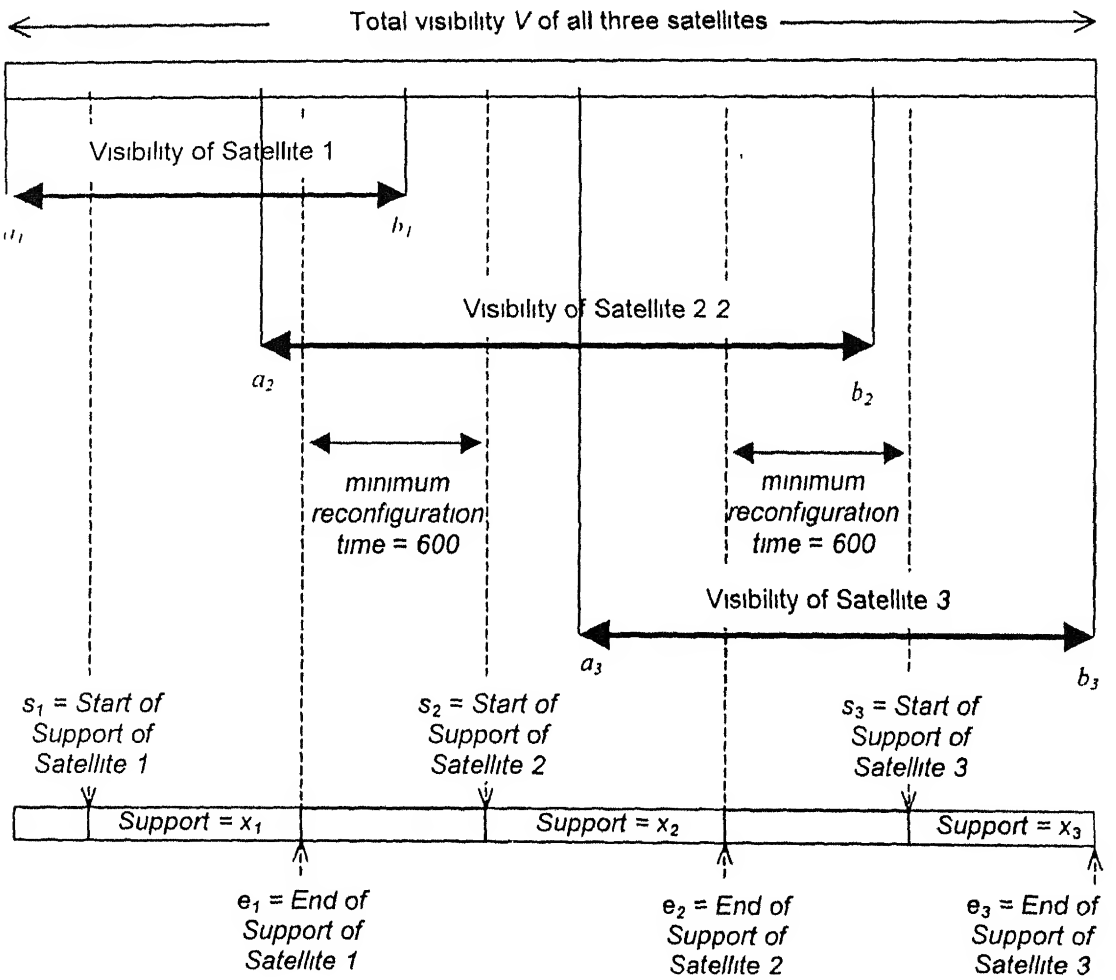
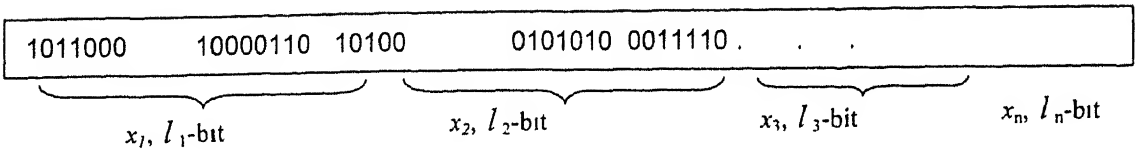


Figure 1.3 - (a)

Figure 1.3-(b)



The Chromosome

Table 1.1:

Scenario 1	s_i	e_i	Support	Total
$C_1 = 1$	616520	616520	0	4670
$C_2 = 2$	617244	617613	369	
$C_3 = 3$	618213	618213	0	
$C_4 = 4$	618213	619196	983	
Scenario 2				
$C_1 = 4$	616520	617115	595	4467
$C_2 = 3$	617715	618318	603	
$C_3 = 2$	617712	617712	0	
$C_4 = 1$	618918	619196	278	
Scenario 3				
$C_1 = 1$	616520	616520	0	4670
$C_2 = 4$	617244	618227	983	
$C_3 = 3$	617712	617712	0	
$C_4 = 2$	618827	619196	369	
Scenario 4				
$C_1 = 1$	616520	617112	592	4344
$C_2 = 3$	617712	617712	0	
$C_3 = 4$	617712	618650	938	
$C_4 = 2$	618027	618027	0	

Optimal Support Schemes for Maximizing Profits

Satellite Support Periods For Profit Scenario 1

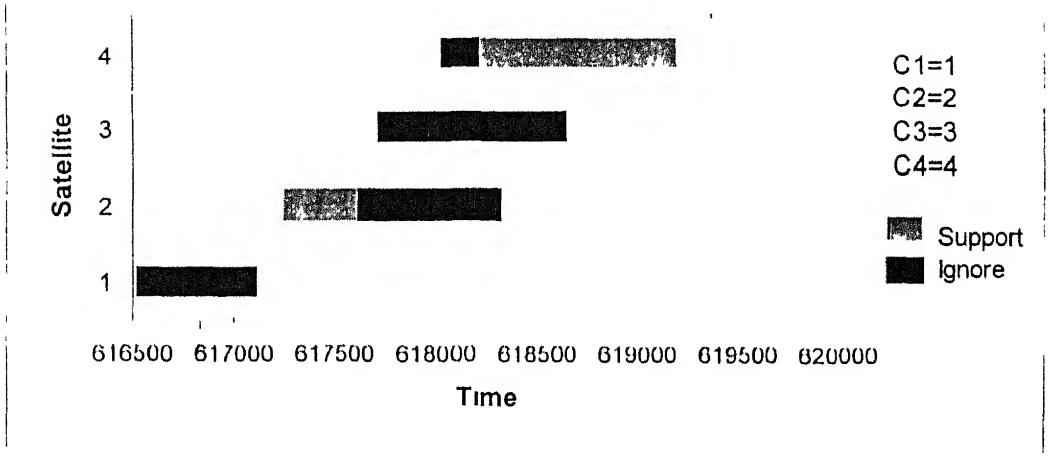


Figure 1.3-(b)

- Satellite scheduling problems involve assigning a resource (or set of resources), a start time, and duration to each task. These problems are usually over constrained. Pemberton and Galiber (1998) have worked on three general types of satellite missions: broadcast, telecommunications, and remote sensing, although a single satellite may encompass more than one mission. They focus on scheduling mission operations, namely the day-to-day activities of an operational satellite These mission operation activities include payload operations (e g , using a sensor on the satellite to collect data), bus operations (e g , maintaining the health and status of the vehicle) and communications operations (e g , transmitting data between satellites or to the ground and receiving information or commands from a ground station) They view the satellite-scheduling tasks as a class of Constraint Satisfaction Problem (CSP) A CSP is typically defined as a set of variables, a set of values for the variables, and a set of constraints on the values that can be assigned to variables Their basic approach can be described by two high-level steps: *first step is the translation of problem into a CSP, and the second step is to solve the CSP* Four basic objects are used to model the problem – tasks, resources, events, and constraints. *Tasks* are the activities and operations to be performed. *Resources* are the people, satellites, sensors, communication channels, etc that are required to accomplish tasks *Events* are used to capture domain-specific occurrences that restrict when tasks can be scheduled (e g., satellite visibility windows). *Constraints* are further restrictions on when tasks can be scheduled that are due to interactions with other tasks or to resource capacity and availability.

A task is scheduled by assigning it a resource, start time and duration, which satisfy all of the relevant constraints; a scheduled task is a tuple

(task, {resource set}, start time, duration)

There are three types of satellite scheduling constraints: task constraints, resource constraints and event constraints. Task constraints are constraints between tasks. For example, a user may require two observation tasks to occur at the same time. Resources also constrain the set of acceptable schedules In the first place, resources have specific

capabilities For instance, a task to take an infrared image cannot be performed by a satellite that doesn't have an infrared sensor. Event constraints are used to describe time windows in which a task can be executed. In satellite-scheduling, event constraints result from the fact that satellites must be able to see ground stations to perform a given task

There are certain characteristics of satellite scheduling problems that have their impact on the scheduling task. These are -

- *Event Windows* Satellites orbit the earth and are visible on various ground stations The visibility start and stop times, on the ground stations, define the event windows for a given task-resource pair If a task only requires a single resource, then the task event windows are equivalent to the event windows for the task-resource pair. If a task requires multiple resources (e g. two satellites must both be in contact with a ground station at the same time), then the task event windows are the intersection of the event windows for each task-resource pair.
- *Alternate Resources* In job-shop scheduling, it is commonplace that a task can be completed by more than one resource This is also true for satellite scheduling problems with a twist Because of the fact that no two satellites can share the sane identical orbit and no two different ground stations can share the exact same ground position, alternate resources are not completely interchangeable Cases where resources are interchangeable are for ground-based resources that are not position dependent (e.g. support staff, post-processing computers, etc.).
- *Periodic tasks.* Periodic tasks are common in satellite scheduling problems Common periodic tasks include maintenance checks, downlinking collected data, and periodic collection over a specific region
- *Preemptive tasks* An example of this type of task is-"ground station L must be observed for a total of 30 minutes during the next 24 hours. Each observation must be at least 5 minutes in duration"
- *Renewable resources·* Some satellite resources can be both consumed and renewed. e.g , data storage and battery power. The problem posed by these resources is to find optimal number of resource renewal tasks.

- *Variable length tasks:* Presence of tasks with variable duration further complicates the scheduling problem.

GREAS (Generic Resource, Event, and Activity Scheduler), models this problem as tasks, resources, events, and constraints, and then searches for a solution using heuristic-search with constraint propagation The approach can be summarized as. choose a variable to instantiate, choose a value for the variable, and propagate the implications of the value selection on the remaining variables This process is repeated until all variables have valid values until one of the remaining variables has a null range (i.e. a dead end) When a dead-end occurs, the search algorithm must backtrack by selecting a different value for one of the previously instantiated variable.

- **Search Algorithms for Constraint Satisfaction Problems:**

Backtracking and *constraint propagation* are two basic types of constraint satisfaction search algorithms (Kern, 2000) Backtracking is synonymous with retraction techniques. Backtracking will always find a solution, but may not be very efficient. Constraint propagation is often viewed as a *pre-processing* algorithm in that it can reduce the problem space of a constraint satisfaction problem before conducting search. This reduction occurs by eliminating inconsistent domain values between variables resulting in *consistency sets.* Depending on the problem domain one or more of these consistency sets may actually be a solution to the problem The need to search for a solution is eliminated. Unfortunately constraint propagation does not guarantee a solution will be found even if one exists. For sufficiently complex problems constraint propagation is often not enough. In these cases it can be applied to the problem prior to conducting backtracking to eliminate inconsistent domain variables prior to searching These algorithms are described in more detail in the following sections

Backtracking

There are many forms of backtracking, but the most popular and simplest algorithm is *dependency backtracking*. This algorithm tries to instantiate each variable, and for each instantiation a consistency check is done to ensure all constraints are satisfied If all the checks succeed, the next variable is instantiated, otherwise, another instantiation with the next value is done If all possible instantiations for a variable fail then a backtrack is done to the most recently instantiated variable This technique can be used to solve the graph coloring problem -

The algorithm starts by instantiating variable V1 with the value red It then proceeds to instantiate V2 with red. It identifies that the constraint between V1 and V2 (i.e., V1 ≠V2) is violated. As a result it discards the value red as a possible choice and selects green. All constraints are satisfied so it selects V3 instantiate next The algorithm selects value red for V3 and checks that all constraints are consistent. The final variable is selected and given the value red Again a constraint is violated so the next value is selected. This continues until V4 is assigned blue At this point all constraints are consistent and a solution is found In this example the backtracking is used to select a different value for a variable (i.e., remove red and try green). It is often the case that the backtracking algorithm needs to go back to a previous variable because no consistent values for the current variable exist This is where the inefficiency of backtracking may manifest itself In the end if a solution exists backtracking will find it.

- **User-centered Scheduling:**

Perry was tasked with the job of specifying a standard approach to scheduling military airspace usage [POP92]. Through study of the various locations responsible for conducting this scheduling, several facts were

identified. First each area had its own policies and processes for scheduling their respective airspace. Second no standard representation of the scheduling process existed. Most scheduling was conducted with pencil and paper and little auxiliary automation. Rather than trying to capture all of the various policies and procedures that each airspace management location used, emphasis on user interface design of the scheduling system would be the primary means of producing deconflicted schedules. The intent was to add more complex reasoning to the interface as the system matured Human schedulers typically resolve conflicts by generating alternatives, assigning priorities, or trying to negotiate mutually acceptable solutions. Because of the distributed, widely differing scheduling strategies inherent in the overall problem, an aid was developed where the user has an explicit role in the scheduling process. This was preferable to the development of a highly complex, fully automated scheduling system Due to the dynamic rescheduling nature of airspace management, it seemed more effective to provide necessary tools via better user interface mechanisms rather than incorporate explicit knowledge of numerous considerations of the scheduling process. Therefore effort centered on providing useful interface components that facilitate forming and maintaining a schedule regardless of local practices or procedures. The user interface is the scheduler's primary means of establishing a deconflicted schedule It is modeled as an interactive Gantt Chart where horizontal areas represent resources being scheduled and the window is divided left to right by time. This allows the scheduler tofocus on a specific time period, yet gain access to distant areas of the schedule as needed By clicking on an activity duration bar, the scheduler can gain detailed information about the activity.

- **Mixed Initiative Scheduling:**

Chien recognizes there are obstacles hampering the application of scheduling technology to real world problems [SC96]. One of the ways to solve these obstacles is through human involvement in the scheduling process. Scheduling systems need to fit into a wide range of operational contexts Most scheduling tasks cannot be completely automated. Since this is true resulting schedules need to be easily understood so the user can modify them In some cases the user required intimate involvement in the schedule construction process By involving humans the scheduling process can be quicker, higher quality, and easily adaptable to dynamic changes A significant need exists for a natural mode of interaction between the user and the system. It is necessary to have a clear and convenient division of labor and control between the user and the system. Commonly the solution provided by the system must be verified and applied with some human intervention. Therefore an interactive, mixed-initiative schedule construction process is needed. Other desirable system characteristics include support for iterative refinement, subjective adjustments by human experts, and planning for interaction points. Fully functional systems are able to integrate scheduling and re-scheduling to support mixed-initiative interactions.

1.7 Brief overview of work done:

Our work focuses on scheduling telemetry, tracking and commanding operations of Indian Remote Sensing (IRS) satellites. These operations require to be routinely executed and can be performed only when the satellites are visible on Telemetry, Tracking and Commanding network of ground stations, distributed all round the globe. Scheduling these operations is a complex task as it involves myriad set of constraints Further, non-linear nature of constraints imparts more complexity to the problem. The multi-satellite-scheduling problem falls under the class of NP-hard problems

In this chapter (Chapter 1) we started with the preliminaries - remote sensing, Indian Remote Sensing satellites and their applications in various fields, ground trace of these satellites, and ground stations. We also discussed that a visibility window of a

satellite on any ground station is characterized by two delimiters - Acquisition Of Signal (AOS) and Loss Of Signal (LOS). Various types of operations are required to be performed on the satellites and these have been categorized as *TTC* operations, *payload* operations and *command* operations All of these operations are done in visibility windows of the satellite Much work has been done in field of satellite-scheduling, a lot of it has been summarized at the end of this chapter.

Chapter 2 explores the telemetry, tracking and commanding operations of the LEO (Low Earth Orbiting) satellites. Each satellite has its own set of operations to be performed on it and these operations have different periodicities tied to it. These operations are performed on the satellites according to a weekly schedule provided to all the ground stations of the network A lot of constraints exist- those imposed by ground stations, satellites, operations, and many more The chapter discusses all of these constraints These are to be considered to solve the TTC operations' scheduling problem in multi-satellite environment. We also discuss in this chapter about the objective of the problem

Chapter 3 throws light on the high complexity of satellite-scheduling problem and describes limitations of traditional programming approaches - linear programming (LP), quadratic programming (QP) and mixed integer programming (MIP) to solve this problem Further, the strengths of constraint-based approach have been discussed. This approach is a powerful tool for solving large, industrial-scale combinatorial optimization and constraint satisfaction problems. Lastly, the chapter discusses how we have viewed the problem and the constraint based heuristic we have proposed to solve the satellite-scheduling problem.

Chapter 4 is an extension of Chapter 3. In Chapter 3, we model the problem as a set of entities contending to get scheduled in visibility windows. These entities are associated with priority values that changes dynamically Entities are scheduled, picking that of highest priority at a time, scheduling it and propagating the implications. This changes the priorities of remaining entities. Chapter 4 discusses the factors responsible for this dynamically changing priority.

Chapter 5 describes the details of programs we have used to implement the steps involved in our heuristic. Six data-files have been used as inputs that are used by the

programs and all of these have been explained in detail in this chapter. Ten programs, written in Turbo C++ implement the proposed method, details of each of these in terms of inputs, outputs and the function performed by the program, has also been explained

In Chapter 6 we discuss the results we arrived at after implementing the method and our conclusions from this study. The data files used by the programs and the code written in C++ for implementing the heuristic has been made available in appendix.

Chapter 7 concludes this study and indicates directions for further investigation.

The term satellite scheduling has been applied to many different aspects of a satellite's operation (e.g , design planning, launch control, lifecycle, etc). In this thesis we have focused on scheduling TTC operations; these operations are necessary for monitoring of health of various sub-systems of a satellite, maintaining its position in fixed orbit, measuring its velocity, etc These operations are performed at the TTC stations of ground network to ensure a regular contact with each of the Indian remote sensing satellites.

2.1 TTC Operations:

TTC tasks refer to *telemetry, tracking and telecommanding operations.* These operations are performed via exchange of signals between the spacecrafts and TTC ground stations, positioned all round the globe. When a spacecraft is visible at a ground station, the station establishes radio link with the spacecraft only if the visibility window on that station has been scheduled to be supported. A weekly schedule is provided to all the ground stations of the network, which guides the stations to support/not to support the visibility windows and perform operations on satellite.

It is a tough and time consuming task for the planners to prepare this weekly schedule for all the remote tracking stations of network It involves a myriad set of opportunity windows, variety of constraints and several other criterion, like operation priority, satellite priority, etc. This research proposes a method for scheduling these telemetry, tracking and telecommanding operations.

To get the feel of the problem, i e to acquire the details of operations performed at ground stations and the rules governing the scheduling process, we visited the ground stations of Lucknow and Bangalore and interviewed the experts of scheduling and those involved in commanding the satellites.

All the operations performed on TTC ground stations (with their code names in bracket) have been summarized below and more scheduling specific information has been tabulated in Table 2 1

Table 2.1 Operations On TTC Stations

Notations used

AELGP=>All Eligible Passes, DAELGP=> Day All Eligible Passes, DFELGP=>Day First Eligible Pass; NFELGP =>Night first eligible pass

.DNFELGP =>Day Night First Eligible Pass,

NLELGP =>Night Last Eligible Pass, O1→O2=>O1 precedes O2

*No minimum=> the command requires a fraction of second

*Cycle => repetetivity cycle of spacecraft

cycle time (in days) is decided by planners

*Sister stations set of stations from where a satellite may be visible simultaneously.

Operation No.	Operation Name	Satellites Affected	Minimum Duration	When Performed	Other constraints	Independent/ Dependent (I/D)
1	TM	All eight	Specified by satellite	11A once every day	Minimum elevation, specified by ground station	I
				IP2. twice every day		
				Others: AELGP		
2	TC	All except IP2	No minimum	AELGP		D TM→TC

3	**TR**	All eight	No minimum	I1A: thrice every day	Only from one sister station;	D TC→TR
				IP2: twice every day		
				Others: AELGP		
4	**PB**	All except IP2 and SC2	Specified by spacecraft	I1A: once every day	not with DW or RawSS or SPS_RT *	D TC→PB
				Others once every orbit		
5	**PYS**	I1B, I1C, I1D, IP3, IP4	No minimum	Twice every day, DNFELGP		D TC→PYS
6	**DTG1TST**	1B	10 minutes	Once a cycle, night pass	Along with DW	D TC→DTG1TST
7	**DW**	All except SC2,	No minimum	On request	Not with PB	D TC→DW
				SC2.once every day	Not with RPA or GRB or PB	

		I1B		I1B with DTG1TST always also on request	Not with RawSS or PB	TC→DW
8	SS (RawSS)	I1B	No minimum	Once a cycle, pass	Not with PB or DW.	D TC→RawSS
9-(i)	SSP_ON	I1B	No minimum	DFELGP		D TC→SSP_ON
9-(ii)	SSP_OFF	I1B	No minimum	NLELGP		D TC→SSP_OFF
10	VHF_TC	I1A, I1B	No minimum	Once a cycle, DFELGP		D TM→VHF_TC
11-(i)	SPS_PB	IP4	2.5 minutes	Once every day, NFELGP	Only over BLR; not with SPS_RT	D TC→SPS_PB
11-(ii)	SPS_RT	IP4	No minimum	Once every day, morning pass	Only over BLR; not with SPS_PB,	D TC→SPS_RT
12-(i)	DCS_ES_ ON	IP4	No minimum	Once a cycle, DFELGP		D TC→DCS_ES_ON
12-(ii)	DCS_ES_ OFF	IP4	No minimum	Once a cycle, DFELGP	DCE_ES should remain ON for around 48 hrs	D TC→DCE_ES_OFF

13-(i)	**PYS_RST**	I1C, I1D, IP3, IP4	No minimum	Once a cycle, night pass		D TC→PYS_RST
13-(ii)	**CS_RST**	I1C	No minimum	Every Monday, night pass		D TC→CS_RST
13-(iii)	**ELP_RST**	I1D	No minimum	Once a cycle		D TC→ELP_RST
13-(iv)	**PYS_TMR_RST**	IP4	No minimum	Every Monday, night pass		D TC→PYS_TMR_RST
14	**OM**	all eight	-	On request	TC must be scheduled at all stations in that cluster. at least one TTC station in next pass	D TC→OM
15	**RPA**	SC2	No minimum	Twice every day once in day time, once in night time	Standing request; not with GRB, only over BLR,LK2,MAU.	D TC→RPA
16	**GRB**	SC2	5 5 minutes	Once every day	Not with RPA, only over BLR	D TC→GRB

2.1.1 Telemetry (TM):

Spacecraft controllers on the ground rely on the telemetry to monitor the health and configuration of a satellite. There are a large number of health parameters to be monitored, defined for various subsystems of the satellite – power subsystem, thermal subsystem, attitude and orbit control subsystem (AOCS), thrusters, etc. The health monitoring equipments in the satellites are always ON, whether the satellite is visible or not from any of the TTC stations. When the satellite is visible to a TTC station, telemetry data is received by the station in S-band.

- This operation confirms the link between a ground station and a spacecraft and is therefore a prerequisite to all other TTC operations.

2.1.2 Telecommand (TC):

Tele-command supports remote commanding in real-time during ground-station visibility. Provision exists for on-board time tagging of telecommands in some satellites (IRS-1D and IRS-P4). Telecommanding is an uplink operation . This command cannot be performed on a satellite from two stations at a time; i e it should be scheduled for a satellite from only one ground station even if the satellite is visible from two ground stations at a time.Also it is actually the human controller who gives a real control signal. Presently following combinations of satellites are being controlled by their single human controllers. For example,

- IRS 1A and IRS 1C
- IRS-P2 and IRS- P3
- IRS-1B and SROSS

Situations arise when the satellites controlled by a single controller are visible over the same or different ground-stations at the same time. In this situation only one satellite should be scheduled to be supported by the controller and also, sufficient time should be

provided to the controller to switch support from one satellite to the other The decision requires to be automated by the scheduler.

2.1.3 Tone Ranging (TR):

This support is provided during ground-station (TTC) visibility in order to find out the position of the spacecraft with respect to the station. Different tones are sent to the spacecraft from ground-station and from the reflected tones, the ground-station extracts the information about the range (distance of spacecraft from ground station), range-rate (rate at which the range is changing, which in turn gives velocity of the satellite), and antenna angles (elevation and azimuth) with respect to the ground station. In this way, both uplink and downlink are necessary to perform this operation

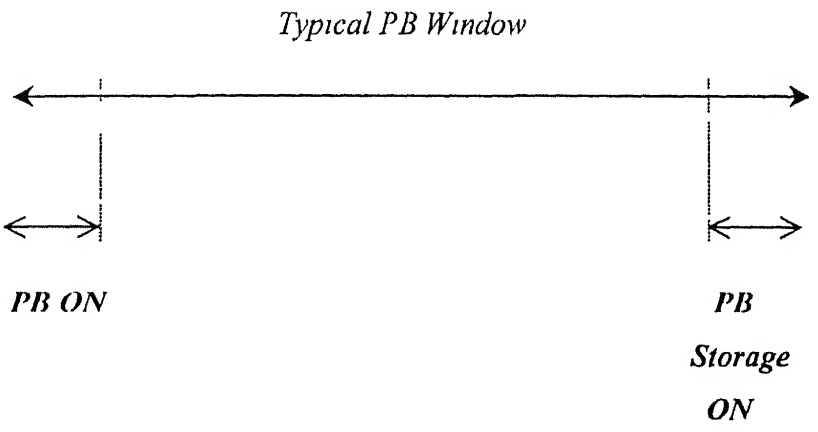
- TR must be done on a spacecraft from a single ground station at a time. That is, during inter-station clash, only one of the stations will have TR scheduled for that satellite
- Also, during inter-station clash, whichever station has been scheduled for TR, same station must be scheduled for other operations like POSIT, PYS, etc. because all these operations require uplink (that is, antenna will have to be switched on only once for all these).
- Although TR is done on a regular and cyclic basis, there can also be a request for TR operation also. For example, there are frequent requests from the SHK station for TR operations. Each TR request specifies the satellite, station, date, orbit and path number.

2.1.4 Playback (PB):

Continuous health monitoring of a satellite is very important. That is, health parameters of all the subsystems of a satellite must be monitored throughout its motion; whether the satellite is in visible or in non-visible region When the satellite is in non-visibility region, its health-data cannot be received by any TTC station Therefore an on-board-memory-device is housed on a satellite with a capacity of storing the health-

data covering the period of one full orbit (approx. 103 minutes). A typical playback command widow comprises of 'playback ON' in its leading end and 'playback storage ON' command at its trailing end 'PB ON' command is given to extract health data from the on-board health- monitoring device. 'Playback storage ON' command is given in *continuous* or *sample mode,* commanding the satellite's on-board health monitoring device to store health-data in continuous/sample mode Sample mode is used when the schedule is not able to allot one PB to every orbit The following figure shows a typical playback window' -

Figure 2.1: A Typical PB Window



2.1.5 Precision Yaw Sensor (PYS):

Due to environmental and internal torques, stabilization of the satellite is continuously affected. If not controlled, shifted image is taken by the satellite. PYS is one of the attitude sensors mounted on spacecraft and senses Sun position. Its location on satellite is such that sunrays fall on it only at two points of its orbit - at North Pole and South Pole positions As the position of the Sun with respect to earth changes everyday, therefore in order that yaw-error can be calculated correctly, the reference value for calculating yaw error has to be changed everyday. This reference value is set twice in a day when sun is visible to this sensor. This reference value is set with the help of this command and hence this command has to be given twice a day

2.1.6 Dynamically Tuned Gyro Test (DTG1TST):

This operation, as name suggests, is performed to do the testing of dynamically tuned Gyro. In IRS-1B there are two Gyros and this operation is done to confirm the performance of the first Gyro

2.1.7 Dwell (DW):

The telemetry system collects the house keeping data from each subsystem and then formats and modulates it on the sub-carrier. There are two formats, viz., dwell and normal. Dwell mode and normal mode formats can be simultaneously received. Dwell mode provides real-time health data at faster rate and is used for health parameters, which require more frequent health monitoring, viz , attitude loss, battery condition, etc

2.1.8 Raw Star Sensor (SS):

This is a sensor boarded on IRS-1B, used to get the attitude information of the spacecraft by sensing the position of the stars accurately and giving their angular position with respect to satellite. This is the most accurate sensor amongst the sensors being used for the same purpose. The word 'raw' here implies that spacecraft gives each and every data obtained (about stars), without processing it. This raw data when received, in real time, by the ground station is processed at the ground station and by cross comparing it with the processed data of star sensing processor, the performance of star sensing processor is assessed. It has been observed that more stars can be seen when spacecraft is in southern hemisphere, therefore this operation is scheduled preferably over MAU and BLR during night passes. As the purpose of this operation is only to confirm the performance of the SSP it is sufficient to give this command once in a cycle.

- If PB or DW has been allocated to any of the sister stations, Raw SS should not be scheduled on any of the sister stations.

2.1.9 Star Sensing Processor ON/OFF (SSPON, SSPOFF):

Star sensing processor is an on-board processor on IRS-1B, used for processing the raw data sensed by Raw Star Sensor When SSPON command is given, the processed form of raw star sensor data is received with TM data.

2.1.10 Very High Frequency Telecommand (VHF_TC):

In earlier satellites (IRS-1A and IRS-1B), in order to have a standby system for commanding the spacecraft in case of S-band failure, very high frequency commanding is made. The special antennas for this mode of commanding are to be handled manually.

2.1.11 Satellite Positioning System (SPS):

Data from this payload, on-board IRS-P4, can be received in real time as well as can be played back using SPS_RT and SPS_PB command The SPS_PB is similar to the telemetry PB operation

2.1.12 Dual Conical Static Earth Sensor ON/OFF (DCS_ES_ON, DCS_ES_OFF):

There are some redundant systems on the spacecraft, which are required to be switched ON/OFF regularly in order to confirm that they are working. This command operation is one of the examples.

2.1.13 Precision Yaw Sensor/ Conical Sensor/Elapsed Orbit Timer-Reset: (PYS_RST, CS_RST, ELP_RST, PYS_TMR_RST)

These command operations are used for resetting above devices of different spacecrafts

2.1.14 Orbit Manoeuvre (OM):

This operation is carried out on request whenever a shift (fall or rise) in the orbit of spacecraft goes beyond tolerance limit, that is, when shift in ground trace goes beyond ± 1 Km. from normal value. The request comes like this: -

Date	Orbit No.	Station	Operation
5-1-2000	1766	BLR	OM

2.1.15 Retardation Potential Analyzer (RPA):

This is a payload on-board SROSS being used for different scientific themes, ionospheric irregularities at equatorial regions, deviation of density and temperature models, study on solar flare and geomagnetic disturbances.

2.1.16 Gamma Ray Burst (GRB):

Stars when burst, emit gamma rays. These bursting stars are recorded by this payload mounted on SROSS. Till date 40 gamma ray bursts have been recorded.

2.2 TTC Ground Stations:

After talking to the persons involved in scheduling at ground stations, we dug out some more relevant pieces of information regarding capabilities of different TTC stations. We found that capabilities of various ground stations are different in terms of working hours, number of antenna (chains) a station possesses, types of operations that can be performed on each chain and minimum elevation angle required for an operation to be performed on a chain. Table 2.2 summarizes these details A chain of any ground station should be scheduled to support only one satellite at a time, further a sufficient delay should be provided to the chain to switch support from one satellite to another.

Table 2.2: Capabilities Of TTC Ground Stations: (contd on next page)

Station Name	Chains	Satellite Supported	Operations	Min. elevation (degree)	Working Hrs.
Bangalore	BLE, BLR	All except IRS-1A,P2	TM	2	00.00 TO 24 00
			TC	3	
			TR	7 5	
			PB	5 5	
			RPA		
			GRB		
			DW		
			PYS		
			DTGITST		
			CS_RST		
			SPS_RT		
			SPS_PB		
			DCS_ES_ON		
			DCS_ES_OFF		
			MSMR_RT		
Lucknow	LK1	IRS-1B, SROSS	TM	2	00 00 TO 24:00
			TC	3	
			DW		
	LK2	All eight	TM	2	
			TC	3	
			TR	7.5	
			PB	5 5	
			DW		
			RPA		
			PYS		
			CS_RST		
			VHF_TC		
Mauritius	MAU	All except	TM	5	

		IRS-1A, IRS-P2	1C	5	15 00 1O 21:00
			TR	7 5	
			PB	5 5	
			RPA		
Bearslake	BR1, BR2	All except IRS-1A, SROSS	TM	5	00 00 TO 24 00
			TC	6	
			PR	7 5	
			PB	6	
			C'S RST		
			ELP_RST		
			PYS_RST		
Shrihari-kota	SH1, SH2	IRS-1A, IRS-P2	1M	2	04 00 1O 11 00
			TC	3	
			TR	7 5	
			PB	5 5	
Biak	BIK	IRS-1C, IRS-1D, IRS-P4	1M	2	00 00 1O 14 15
			TC	3	
			PB	5	
PortBlair	PBR	IP4	1M	3	04 00 1O 8 00
			TC	5	
Wheilhem	WHM	IRS-P3	1M	5	00 00 1O 24 00
			TC	7	
			TR	10	
			PB	10	

*Above table shows that the two chains LK1 and LK2 of Lucknow ground station are not equally capable

2.3 Types Of Constraints Encountered·

A CONSTRAINT restricts the set of values assigned to a variable CONSTRAINTS restrict the assignment of start and end times and the allocation of RESOURCES to ACTIVITIES

From the details of operations, Table 2.1 and Table 2.2, we could infer variety of constraints: -

- **Value-compatibility constraints:** Value-compatibility constraints ensure that the right types of resources are assigned to a given activity. In TTC operations' scheduling scenario, examples falling into this category of constraints are -

  Chains of stations (same/different) are different in their capabilities A chain can support a specified set of satellites The chains also differ in terms of types of operations it can perform

- **Causal Restrictions:** constitute another category of constraints They define what conditions must be satisfied before initiating an operation. In our problem, examples of causal constraints are -

  Precedence: A precedence constraint on an operation states that other operation must take place before (or after) it Examples of this category are. -

  There should be around 48 hours gap between DCS_ES_OFF and DCS_ES_ON,

  A single observation usually requires a sequence of several smaller steps to assure successful data collection. monitoring health of the satellite (telemetry links), pointing, tracking, etc (TM→TC, TC→VHF_TC, ..).

  Resource Requirement These are another causal constraint that is the specification of resources that must be present before or during the execution of a process. Example· -

  For an operation to be performed on a chain, the angle of elevation of the satellite at that chain should be greater than or equal to a minimum value.

- **Temporal Constraints:** Temporal constraints restrict the values of time-sensitive variables. An example is the due-date constraint of a demand The due-

- date restricts the end-time of the last activity selected to satisfy the demand In context of TTC scheduling, an example of this constraint is -

  A operation can be done on a satellite only in its visibility window, that is when the satellite is visible from a station of ground network, the execution time should be such that the operation gets performed before the visibility window ends.

- **Availability Of Resources:** Resource-availability constraints model resource capacities as constraints (Kern, 2000). That is a resource of capacity 2 is constrained to support no more than two activities at a time. As resources are assigned to specific operations, it may be unavailable for other uses during some time period. Resource availability is also generated by work shifts (working hours of stations) and down times (station is under maintenance, therefore not available) In our problem this category of constraints have a major role and are termed as *controller clash, visibility clash* and *inter-station clash.*

  Controller Clash: Operations are performed on a satellite (a satellite support can occur) when a line-of-sight communication link is established between a satellite and a ground remote tracking station to transfer electronic commands up or electronic data down These electronic commands are given by a human controller, i.e. every satellite has its human controller. A single controller can be assigned to take care of two satellites - give commands to satellite when the satellite has its visibility window over any ground station It happens many times that visibility windows of both of the satellites in control of a single controller occur at the same time (overlapping windows) and the controller is in dilemma regarding which satellite he should support This is to be automated by the scheduler. A controller requires some time to switch support from one satellite to another, this time is called *preparation time.*

  Visibility Clash: A satellite mission involves a number of satellites It happens very frequently that more than one satellites are visible on a single station (antenna) at the same time - some portion of their visibility windows is overlapping. One antenna can only follow one satellite at a time. So the scheduler should assign an antenna to support only one satellite at a time.

Further, to switch support from satellite to another, a certain delay, called as *reconfiguration time*, should be allowed to an antenna (for reconfiguration before being able to track another satellite).

Inter-station clash: A satellite may be visible from two different chains at a time Each spacecraft requires to be up-linked from one chain at a time.

- **Other Constraints:** We encountered several other constraints, such as every operation of each of the satellites requires a minimum duration (in minutes) to be performed;

each spacecraft has a set of operations that require exchange (between satellite and ground station) of *same carrier frequency signals*, therefore, not more than one operations of this set can be scheduled at the same window, all satellites must be regularly tracked with a minimum duration for telemetry, etc

The primary property of a CONSTRAINT is whether or not it may be modified or violated. The problem solver is never allowed to violate *hard constraints* while *soft constraints* are considered relaxable if need be. The designation of *relaxable constraints* is typically accompanied by a specification of *objectives* or *preferences* When due dates can be relaxed, minimizing tardiness is a common objective Objectives and preferences prioritize the space of possible relaxations of a CONTRAINT and provide a basis for measuring *solution quality* (Kern, 2000)

2.4 Problem Modeling:

The tabulated details of operations and station-capabilities, give a generalized view of the problem We may informally describe visibility-scheduling (over TTC stations) problem as follows: -

- Given a set of satellites of the constellation to be tracked, the set may contain any number of satellites;
- Given a set of TTC ground stations performing operations on these satellites;
- Each of the ground stations may have greater than one tracking ***chain*** (independent set-ups used for establishing radio link with satellites) available to support the satellites' operations,

- Given reference time interval (=1 week) in our problem;
- Given a set S of visibility windows corresponding to each of the satellites on different ground stations on the reference time interval, a window may meet/may not meet the requirements for an operation of a satellite to be performed at that window (minimum duration required by an operation to be performed, minimum elevation required for an operation to be performed on a particular ground station...),

Objective: The problem is to schedule the *maximum number of operations* of satellites on TTC ground stations, providing minimum required coverage at each of the windows for the operations scheduled at that window, using a subset S' of S visibilities which is admissible (these visibilities meet all hard constraints)

Constraints:

- Each spacecraft has its specific operations, periodicities of these operations are different and the operations may be specified to be scheduled in morning/night; each operation requires a minimum duration to be performed, an operation may have some preceding operations (more operations to be done, before the operation is performed). Figure 2.2 on next page shows the periodicities of the operations.
- From among the operations on a satellite that require exchange (between satellite and ground station) of *same carrier frequency signals*, not more than one operations of this set must be scheduled at the same window.

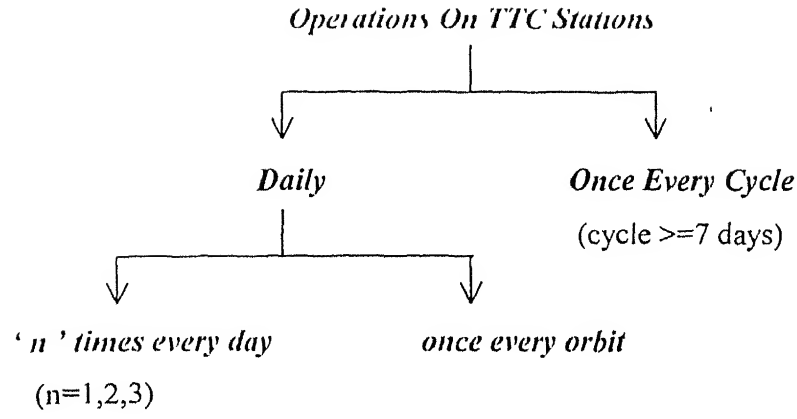


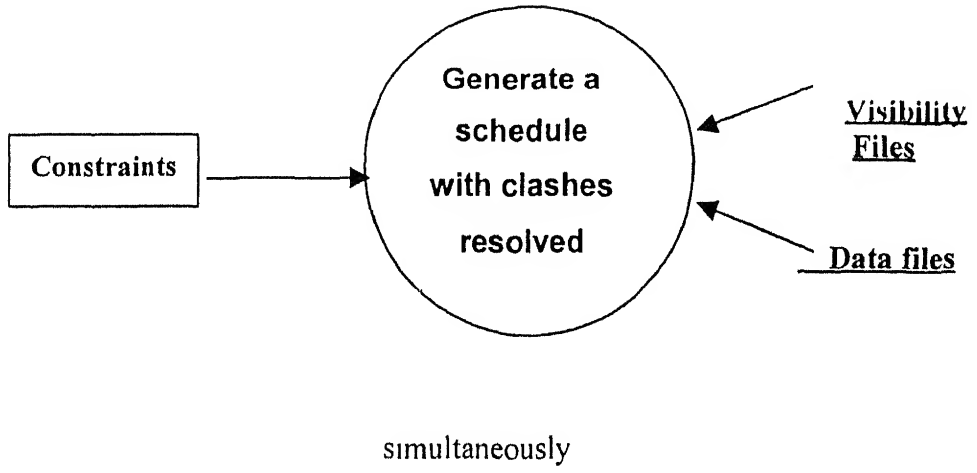
Figure 2.2: Periodicities of TTC Operations

- Each spacecraft has its human controller, two spacecrafts may have single human controller. The human controller, having two satellites under his control, can support only one satellite at a time, further he requires some *preparation time* to switch support from one satellite to the other.
- Each chain is characterized by - set of satellites that it can support, a defined set of operations that it can perform, minimum elevation (of the satellite on the chain) required for each of the operations it can support, and a reconfiguration time (=time to switch support from one satellite to another).
- A satellite must be tracked from only one antenna at a time
- A TTC ground station may be available for service (support operations on satellite) for all 24 hours in a day, while others are available for a limited time over a day;
- All satellites must be regularly tracked with a minimum duration for telemetry. This regular interval for tracking is determined by the planners of the mission. For new satellites (much part of their service life is left), a gap longer than 3 orbits (around 300 minutes) is not tolerable For older satellites, which are towards the end of their mission life, the regular interval at which support is required is larger - once or twice or thrice per day.
- Depending upon the phase of satellite and operational importance, every satellite is assigned a priority; high priority is given to satellites dedicated to global payload missions, lower priority is given to satellites taking payloads at limited stations

- only, and lowest priority is given to satellites that are operationally less important. Satellite priority and operation priority must be considered.

It can be concluded that the problem is a complex one, its complexity grows at least exponentially with the problem size (number of satellites, number of chains, number of operations, etc) Exhaustive enumeration is not possible Most of the constraints are *hard* (cannot be relaxed) Automation of decisions is time-consuming A scheduler should be able to allocate the tasks on the satellites, satisfying all these constraints. The input parameters-number of satellites; number of operations, an operation is independent or dependent; reconfiguration time, number of stations and its chains, etc, all these should not be frozen but should be read from an input file.

Figure 2.3: Data Flow Diagram For The Scheduling Problem



Visibility files are prepared by ISRO on weekly basis for each of the spacecrafts Each record of a visibility file contains satellite visibility information – date, satellite name, station name, orbit number, maximum elevation, AOS, LOS A sample of records from visibility file of I1D has been shown below -

```
2000  1 10 IRS-1D  BIK  11931  4 275   0  3 46   0 11 49
2000  1 10 IRS-1D  ASA  11931 11 484   0  8 28   0 20 04
2000  1 10 IRS-1D  FAK  11932 14 961   1 17 42   1 29 34
2000  1 10 IRS-1D  SEK  11932 18 745   1 30  4   1 43  3
```

In the previous chapter, we discussed telemetry, tracking and commanding operations of satellites and the various categories of constraints specific to their scheduling. In this chapter we discuss the methodology we have used to schedule these operations We will start with describing the complexity of the problem and why traditional techniques of optimization – mathematical or integer programming, etc. are not suitable to solve this kind of problem.

3.1 Complexity Of The Problem:

One way to find an optimal solution is to exhaustively search all the possibilities and to select the best solution compatible with the chosen metrics of performance. This approach is not suitable for scheduling operations of satellites in multi-satellite scheduling problem. Like the job shop problem which is NP-complete, the number of possible options in this problem is very high, therefore, the process of exhaustive enumeration will be very time consuming, if not impossible. To illustrate the complexity we will take an example

Example 3.1: This illustrates order of complexity of satellite-scheduling problem

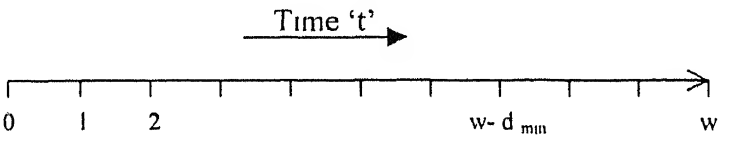
Let us consider a visibility window of width 'w'. Time is divided into discrete into units. If an operation, for which the minimum duration required to perform is d_{min} and maximum duration is equal to total width of the window 'w', is to be performed in this window, then number of scheduling options comes out to be (Wolfe and Sorenson, 2000)

$$(w-d_{min})^2 / 2 + 3*(w-d_{min}) / 2 + 1$$

The "1" occurring in above expression, accounts for the option of *not* scheduling the operation From the expression it can be seen that number of scheduling options on a

window grow exponentially, in a 'n' window scenario, i.e in a problem involving 'n' windows, there will be w^{2n} scheduling options Thus, exhaustive enumeration strategy is not suitable for the scheduling-problem, for it will be very time consuming

Calculation Of Number Of Scheduling Options:



Possible Starting Points	Number Of Possible Options
t=0	$w-d_{min}+1$ = 'k' (say)
t=1	k-1
t=2	k-2
.	.
.	.
.	.
t= w- d min	1

Options = $\frac{k(k+1)}{2}$

Simplifying,

Options = $(w-d_{min})^2 / 2 + 3*(w-d_{min}) / 2 + 1$

$= \mathbf{O}(w^2)$

Other considerations (ILOG Optimization Suite: White Paper, April 1998) are as follows:

- Scheduling problems require the introduction of additional variables (satellites, chains, operations). This would make the problem more complex.
- The linear programming, quadratic programming (QP) and mixed integer programming (MIP) technologies are well understood and widely used throughout the operations research and math programming communities These approaches have been tremendously successful in solving a broad range of commercial

- resource allocation problems Linear and mixed programming algorithms are unquestionably powerful, useful and vitally necessary in optimisation, unfortunately, these approaches also have some limitations
    - Certain classes of problems, like large scale scheduling problems, problems with a high level of symmetry, problems with a significant number of logical relationships, and certain classes of assignment problems remain difficult for LP and MIP technologies. MIP algorithms are not always ideal for applications in which users want good, feasible answers rather than proven optimality.
    - Traditional techniques for modeling LP, and MIP problems put a heavy burden on users to express the model. Users must express all problem constraints in terms of strict linear relationships Logical expressions must be expressed indirectly using non-intuitive modeling techniques.
    - Users of LP and MIP technologies have limited means to guide the solution search with their own knowledge of the problem
- A typical problem with as set of 870 initial windows, when modeled as linear programming in 0-1 numbers, led to more than 380, 000 constraints The only preprocessing took more than one hour on a SUN SS30 workstation and was very long to solve

In our problem. there are around 1800 windows (after filtering out those falling outside the working hours of ground stations or that are not supportable because of satellite- supportability-on-chain constraint).So, the complexity can be guessed A method is required to solve the problem that is flexible enough to incorporate any number of satellites, chains and operations and can give an acceptable schedule

3.2 Constraint Based Approach:

The constraint programming (CP) paradigm has emerged (web reference 4) in the last decade as a powerful tool for solving large, industrial-scale combinatorial optimization and constraint satisfaction problems. While constraint satisfaction problems typically involve searching for a single solution (i.e., a valid assignment of values to each

of the decision variables that satisfies all the constraints), augmenting the framework with *objective functions* (such as cost functions to be minimized or profit functions to be maximized) provides a powerful technique for solving combinatorial optimization problems. Thus constraint-based techniques are nowadays routinely used for solving problems such as job-shop scheduling, timetabling, crew rostering, resource allocation, vehicle routing, logistics management, design and configuration

- Constraint programming permits far *greater flexibility* in problem specification than traditional operations research techniques This permits us to model some unique aspects of problem domains that could not be exploited within the relatively rigid representation frameworks of traditional operations research techniques, leading to significant performance gains.
- Traditional formulations of constraint programming problems assume that the input set of constraints is solvable This is a somewhat unrealistic assumption, since it requires users to manually pre-process the input set of constraints to identify and resolve inconsistencies Constraint based approach is more general, it is able to accommodate perturbations in problems and evolving business needs.
- Even when the problem domain places theoretical limits on our ability to generate optimal solutions within *reasonable time bounds*, constraint programming frameworks permit the specification of heuristics that lead to acceptable, near-optimal solutions
- An additional strength of constraint based approach is that these are often *more intuitive.*
- Constraint-based systems *scale well into large problem spaces* and yield result much faster than other techniques

A key innovation behind constraint programming is constraint propagation Propagation is a generalisation of data-driven computation. (web reference 5) A *propagator* is a procedure that examines the existing search state to find commitments that are logically implied by the current constraint graph, but are not explicitly present [CB98]. By making these constraints explicit they are used to prune the number of possibilities to be explored in the search space. A key requirement is that a propagator be sound in the commitments that it makes. A propagator never infers a constraint that is not

a logical consequence of the current problem state The result of making more constraints explicit is that other propagators might then be used to further prune the search space. The amount of constraint propagation that enables a problem-solver to be the most efficient varies with the problem solver, the application domain, and the problem solving context [CL94] Therefore control is required when applying propagators

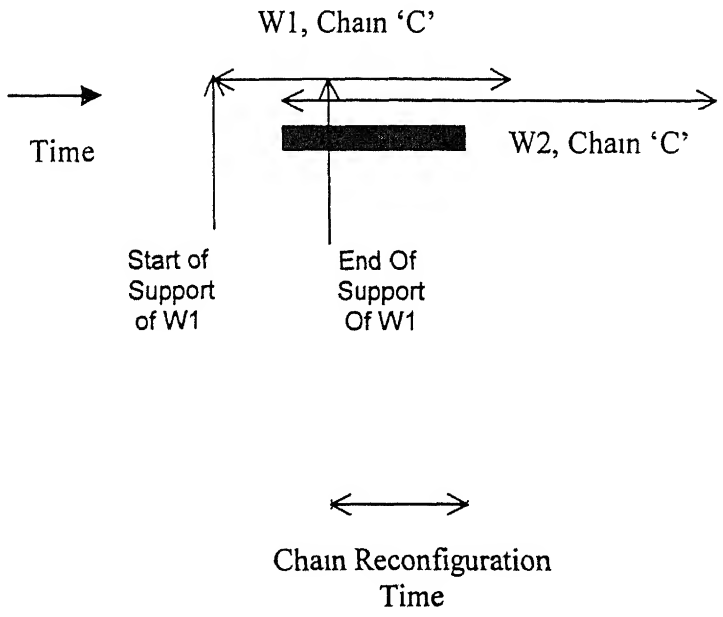
The technique of constraint propagation is helpful in reducing search space, to understand this concept of *constraint propagation*, we will take two examples

e.g. 1: In the Figure 3.1, two visibility windows W1 and W2 occur on same chain, with some part of their windows overlapped If W1 is supported in time interval ' I' on chain 'C'

I=[Start Of Time, End of Time]

then, we can propagate the "one-chain-one-satellite at a time and chain-reconfiguration-time" constraints by removing a part of window from W2 that becomes unusable (any moment of this part can no more be used to support any other operation).

Figure 3.1: Propagating "one-chain-one-satellite at a time and chain-reconfiguration-time" constraints



On allocating interval 'I' on W1 for some operation of the satellite, part of W2 that gets blocked has been represented by shaded area in Figure 3.1.

e.g. 2: A common constraint satisfaction problem is *resource allocation* (Kern, 2000) Resource allocation attempts to assign resources to various tasks This assignment is affected by the capacity of the resources in the domain The meeting problem is a good example of resource allocation Here the "resource" to be allocated is a meeting time. The various "tasks" are the people required to attend the meeting. If Garima, Sagar. and Makarand if are required to attend and the days each can meet are {1,3.4,7}, {1.3.4,9} and {3,4,9.11} respectively, the goal is to find a day they are all available Backtracking can be used to solve this, but much iteration may be required. Instead this problem can be preprocessed using arc-consistency Since a constraint exists that they all must meet on the same day, only days they have in common are considered. Therefore days they do not have in common are removed. The result is the set {3,4} of days that are possible solutions If one of the people did not have any days in common with the others, there would be no solution to this CSP This eliminates the computation required to search for a solution

3.3 Basic "Objects" Of The Problem:

Two general forms of schedule construction exist: *constructive scheduling* and *repair-based scheduling* [CL94] Constructive scheduling attempts to extend a partial schedule until it is complete It checks along the way to ensure that the current constructed schedule is valid This check guarantees that the final schedule is complete and valid. Repair-based methods attempt to iteratively modify a complete, but possibly flawed, schedule to remove conflicts or further optimize a solution An example repair-based method is *iterative repair*. Iterative repair incrementally reschedules in a manner that minimizes changes to the previous schedule. *Iterative refinement* is a constructive scheduling algorithm that schedules activities in an iterative fashion using various techniques to sequence activities based on, for example, criticality or frequency. The method we have proposed in this chapter falls into first category (constructive

methodology). Before going any further, we will discuss the problem, as viewed by the scheduling methodology we are implementing: -

The scheduling problem involves periodic tasks that include maintenance checks, downloading collected data, etc. Each satellite has a set of periodic tasks tied to it The cycle or periodicity of each of these tasks is decided by planners of satellite mission

In a week, a number of cycles occur for a LEO satellite's operation – seven cycles for once every day type operation, fourteen cycles for twice every day type operation, and so on Each of the cycle contains a set of visibility windows. We have termed each of these sets of visibility windows as *slot* and each of the satellite-operation-slot combinations as *entities* In other words, a slot for a satellite's operation is the set of visibility windows, one out of these visibility windows demands to be scheduled for this operation of satellite A satellite's operation may have many slots in the reference time interval (1 week in present problem) and this number is determined by the cycle of satellite's operation. For an operation to be performed once every day, there would be seven slots in one week, for an operation to be performed once every orbit, total number of slots would be equal to number of orbits traversed by the satellite in one week. .. and so on Thus, the total allocation scenario, containing all satellites, all chains and all operations contains large number of entities. These entities are the basic "objects" of the problem.

3.4 Our Approach:

A constraint, in the context of optimization, is not simply a restriction on some values of the decision variables, but the aggregation of a variety of knowledge used in the reasoning process. Thus, constraints have a central role to play in the development of schedule. All the constraints specific to our problem – physical constraints, causal restrictions. resource availability and others, were discussed in previous chapter. These particular constraints are "hard" in nature and therefore none of these can be relaxed. The method we have proposed is a constraint-based approach, it searches for a solution that satisfies all of these constraints and gives a schedule that is then hopefully good enough

How the proposed search process, to identify a feasible allocation advances, using constraint knowledge, may be explained in following steps: -

Step1: ***Compress visibility files applying 'satellite supportability on chains' and 'station working hours' constraint.***

Opportunity windows for scheduling various periodic tasks of the satellites are the *visibility windows* at the ground observation stations positioned all round the globe During its motion round the earth, a satellite crosses a number of observation tracking stations in each of its orbits and all of these are visibilities are available a priori in visibility charts prepared for individual satellites

A typical record of visibility chart would contain the following fields or attributes -

(

year. month, date, name of satellite. name of ground station, orbit number, angle of elevation, AOS hour, AOS minute, AOS second, LOS hour, LOS minute, LOS second

)

The proposed solution procedure starts with implementing following two hard constraints. -

i. Satellite supportability on a chain

ii. Work hours of TTC stations

If a visibility window on a chain falls outside the working hours of the chain, it cannot be supported for any of the operations, so it is rejected Also, if a visibility window of a satellite occurs on a chain that does not support that particular satellite, the window is of no use All such windows are also rejected.

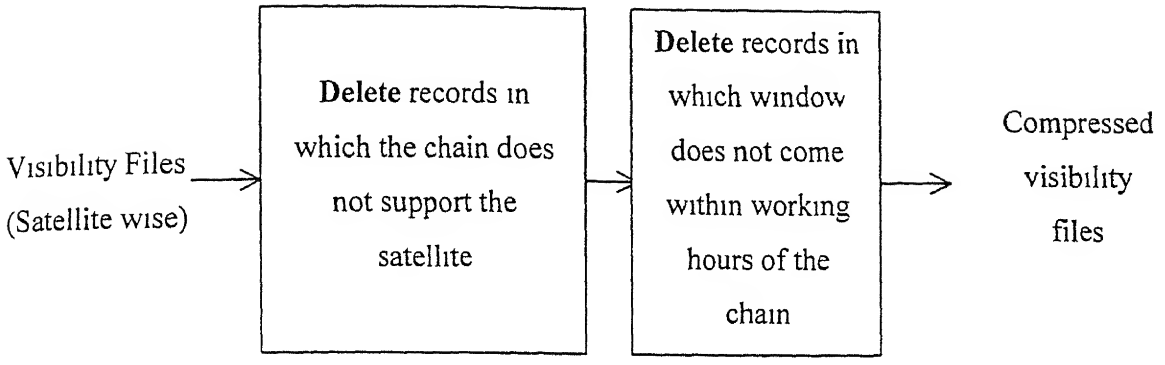


Figure 3.2:
Compressing
Visibility Files

Implementation of above two constraints prunes off the search space The resulting visibility files contain lesser number of windows to be considered for scheduling. Thus, the two constraints have no role left in further process of scheduling- completely move out of the scene.

Step2: ***Maintain status of operations of a satellite as 'p'/'n' on each of the visibility windows of the satellite, applying 'operation supportability on chain', 'minimum duration to perform an operation constraint', 'minimum elevation to perform an operation on a chain' constraint and 'precedence' constraint.***

Having pruned off the solution space for all satellites, the scheduler focuses on possibilities of performing the various operations of a satellite on each of the remaining visibility windows of the satellite At this point many constraints enter the picture·

i. 'Minimum elevation to perform an operation on a chain' constraint
ii. Operation supportability on chain,
iii. Minimum duration constraint on operations
iv. Precedence constraint among operations

On a chain, where a visibility window of a satellite occurs, some of the operations of the satellite may not be supportable due to number of factors A chain may not be able to perform an operation because of angle of elevation lesser than that required to perform the operation on that chain Or, a chain may not designed to support all the operations of the satellites it can see. Also, according to "minimum duration constraint on operations",

each of the operations of a satellite requires a minimum length of time to be performed and hence it demands for a minimum length of visibility window (preemption is not allowed). The length of visibility window (LOS – AOS) may not be sufficient to support some of those operations, making the status of operation 'impossible to perform' at the window Further, any operation of satellite that has passed through these barriers (chain of visibility window can support the operation and length of visibility window is sufficient to support the operation), may remain impossible to be performed at the window because of precedence constraint. Figure 3.3 explains this clearly

Example 3 4

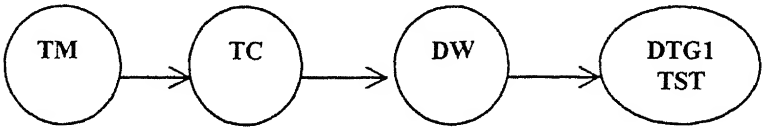


Figure 3.3: A Precedence Constraint

Above figure shows a typical precedence relationship existing among operations, TM, TC, DW and DTG1TST Suppose now that a visibility window of length '7 minutes' occurs on a chain C Each of the above four operations are supportable at chain C. Suppose that the minimum durations required to perform above operations are 8 min, 1 ms, 5 minutes, and 0 5 ms respectively. Applying now, the "operation supportability on chain" constraint and "minimum duration constraint", all of these operations except TM can be performed on the window. However, on imposing "precedence constraint", none of the operations can be performed on this window.

Precedence constraints can be of various types. Some of these types have been shown in Figure 3 4 below In the above example, type of precedence constraint existing among the four operations of a satellite is of type "out-tree"

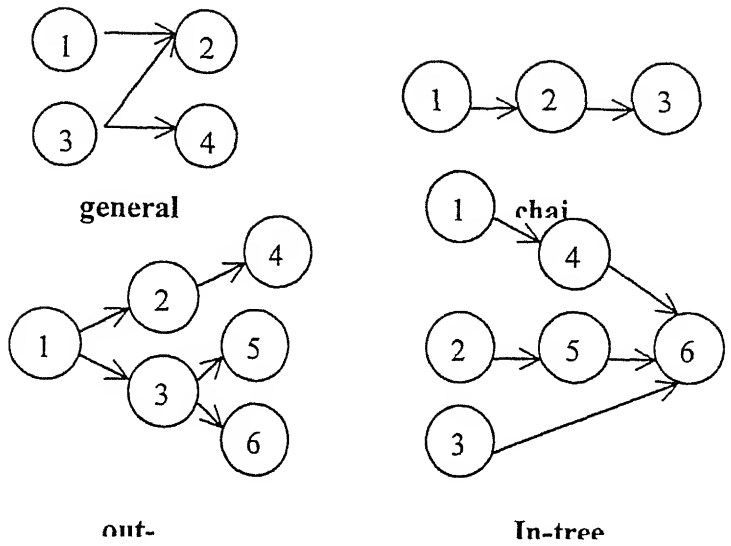


Figure 3.4:Types Of Precedence

Having applied step 2, the scheduler is able to identify at each of the visibility windows involving all satellites, the status of all planned operations of the satellites as "possible/not possible to perform" Thus, each visibility window of a satellite would now contain status of all the operations of the satellite. "Operation supportability on chain" constraint and "minimum duration constraint" would thus have no role left to play further

Step 3: ***Identify entities, that is, satellit e- operation - slot combinations.***

As explained earlier, basic object (termed as *entity* by us) of the problem is :

' Satellite - Operation- Slot'

In this step, the scheduler identifies all these basic objects When summed up for all the satellites for a seven day schedule involving eight LEO satellites and 12 chains, these come out to be around 1300.If an operation of a satellite is of type cyclic (cycle>7 days) and the date on which it is to be performed does not come within the reference time period of one week, there would exist no entities for that satellite – operation.

Step 4:*Schedule operations at confirm windows.*

In this step the scheduler identifies 'no-clash' windows. No-clash window, suppose start of window is AOS' and end of window is LOS', is one that has: -

- Occurred on a chain when no other visibility window (of any of the satellites) occurs on that chain in time period or interval given by,

  [AOS'- chain reconfiguration time, LOS'+ chain reconfiguration time]

  Therefore on scheduling this entire window for support, no part of any of the other visibility windows gets blocked due to the "one-chain-one-satellite" constraint or "chain reconfiguration time constraint" That is, there is no dilemma in deciding whether this window or some other window should be supported by the chain.
- AND, it occurred when no visibility window of other satellite controlled by the same controller occurs in time period given by,

  [AOS'- controller preparation time, LOS'+ chain controller preparation time]

  Therefore on scheduling this entire window for support, no part of any of the other visibility windows of the other satellite controlled by the same controller gets blocked due to the "one-controller-one-satellite" constraint or the "controller preparation time constraint"
- AND, no part of the window is visible from any other chain at this time.

Having thus identified 'no-clash' windows (or, *confirm* windows) in each of the satellite's visibility files, the scheduler schedules all the operations that are possible on these and are *waiting* to be scheduled When this step is implemented many entities get scheduled.

At this stage, in the total set of entities, some are in scheduled state, while the remaining are still waiting to get scheduled. Also, each of the unscheduled windows is clashing with at least with one of the remaining windows of all satellites. To schedule these waiting entities, the scheduler employs "highest priority entity first " approach using dynamically defined priority values of entities. In Chapter Four, we discuss the method for defining dynamic priorities.

Step 5: *Among contending entities, select the highest priority entity, schedule it and propagate constraints. Do this until no entity is left that has some opportunity to get scheduled.*

The scheduler now employs a constraint-directed approach search through solution space and arrive at the final schedule. The search takes place in three levels: -

i Prioritization and entity selection level

ii Allocation level

iii Constraint propagation level

Prioritization and entity selection level: is responsible for selecting the next unscheduled entity to be added to the existing partial schedule. The contending, unscheduled entities are ranked according to:

$$\text{rank}(\text{entity}_I) = \text{priority}_I$$

Priority_I of an entity_I includes two features: -

i Slack, and

ii Slots previously missed

Slack is a measure of number of scheduling options, i e number of windows available for a given entity in its slot Entities with high slack have a large number of options to be scheduled, and conversely entities with low slack have very few ways to be scheduled. If a competing entity has been missed for one or more than one immediately preceding slots, then it priority of getting scheduled should be high This is also taken care of by the priority function, to be discussed in next chapter.

Allocation level: The entity with highest priority is selected (tie is resolved on the basis of satellite priority) and scheduled at the first window in its slot where the operation is possible to be scheduled. Other entities that are waiting to be scheduled, have this window in their slots and are possible on this window are also scheduled, if more than one operations of *same carrier frequency* set of the satellite are possible on this window and entities of these operations containing this window in their slots are not yet scheduled, we choose to schedule the one that has lowest slack value. A minimum support time that is sufficient to support all of these scheduled operations is allocated.

Constraint propagation level: As explained earlier, the key innovation behind constraint programming is constraint propagation The main objective of constraint propagation is to remove *redundant values* from the domain of the variables and remove redundant *compound labels* from the constraints A value of a variable is a redundant value if its removal does not affect the solution of the constraint satisfaction problem (CSP). A compound label is the simultaneous assignment of values to a set of variables [ET93]. That is (<x1,v1>,<x2,v2> . <xn,vn>) denotes a compound label of assigning v1, v2, .. ,vn to x1, x2, .,xn respectively

Propagation of constraints is performed when scheduling decisions taken earlier restrict decisions further on. It helps reduce the search space. We discuss below how we may use this tool to solve the present problem

Having allocated an opportunity window for an entity, we can propagate the following six constraints in the mentioned sequence· -

i 'One-out-of-all same carrier frequency operations' constraint
ii 'Controller clash' constraint
iii 'Visibility clash' constraint
iv 'Inter-station clash' constraint
v 'Minimum duration' constraint
vi 'Precedence' constraint

Propagating 'One-out-of-all same carrier frequency operations' constraint. If more than one operations of same-carrier-frequency-set of the satellite are possible on the window selected to be scheduled, and entities of these operations containing this window in their slots are not yet scheduled, we choose to schedule the one that has lowest slack value. The implication of this step is that the status of other operations of same-carrier-frequency-set becomes "n (cannot be scheduled)" on this window Consequently, the entities of these operations containing this window in their slots, are left with lesser options in their slot.

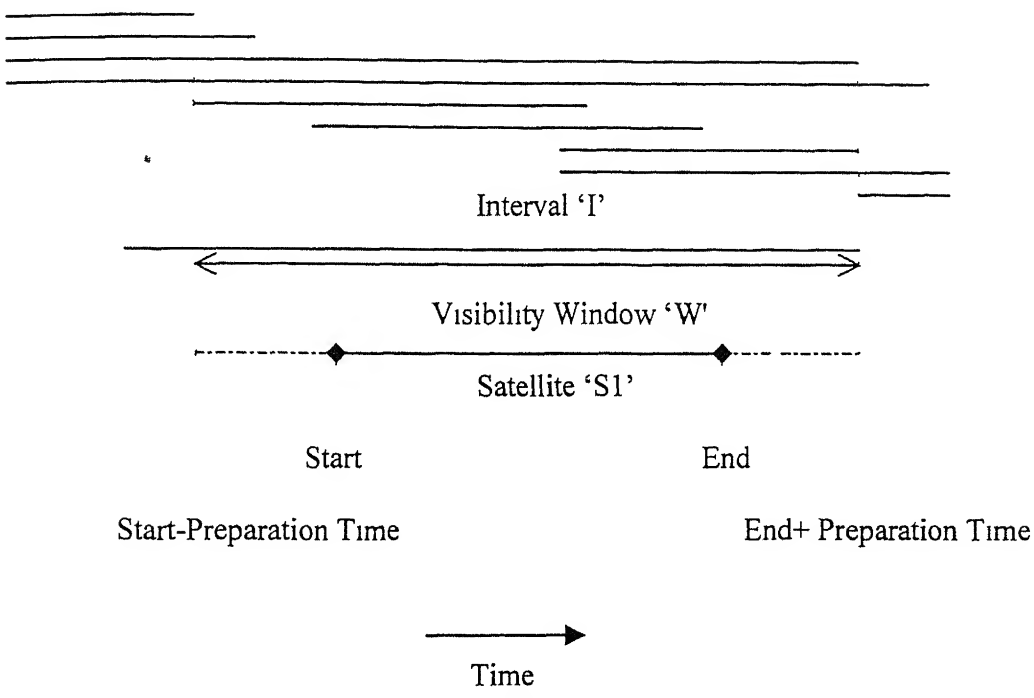
Propagating 'controller clash' constraint: Having allocated an opportunity window for a selected entity, a part of one or more visibility windows of the same controller's other satellite may get blocked – the controller is busy conrolling a satellite and therefore becomes unavailable to support the other satellite he handles. Figure 3 5.1 explains how one may propagate the implication due to the 'controller clash' constraint.

Propagating 'visibility clash' constraint: Having allocated an opportunity window on a chain for a selected entity, a part of one or more visibility windows of other satellites on the same chain gets blocked – the chain is busy supporting a satellite and therefore becomes unavailable to support any other satellite for some time Figure 3 5.2 explains how we may propagate the implication due to the 'visibility clash' constraint

Propagating 'inter-station clash' constraint: Having allocated an opportunity window on a chain for the satellite of selected entity, a part of one or more visibility windows of the satellite on other chains gets blocked – the satellite is being supported from a chain and therefore cannot be tracked from any other chain at the same time. Figure 3.5.3 explains how one may propagate the implication due to 'inter-station clash' constraint.

Propagating 'minimum duration' constraint: Propagation of 'controller clash' constraint, 'visibility clash' constraint and 'inter-station clash' constraint results in shortening of many of the visibility windows or invalidating some windows for supporting any operation. As a consequence, some of the windows become too short to support some of the operations of a satellite In other words, the status of some operations at the shortened visibility windows consequently gets converted from "possible" to "not possible" because length of the window is no more long enough to support an operation that requires a minimum time, that is greater than the new shortened length of the window

Figure 3.5.1: Controller Clash Scenario:



Controller Clash Scenario 'W' is a visibility window of satellite S1 under a controller The same controller has to control another satellite S2. If S2's any of the visibility windows (regardless of the chain on which it is visible) matches with any of the visibility windows shown as light lines in the figure above, then status of window 'W' is acknowledged as a clashing window (controller clash).

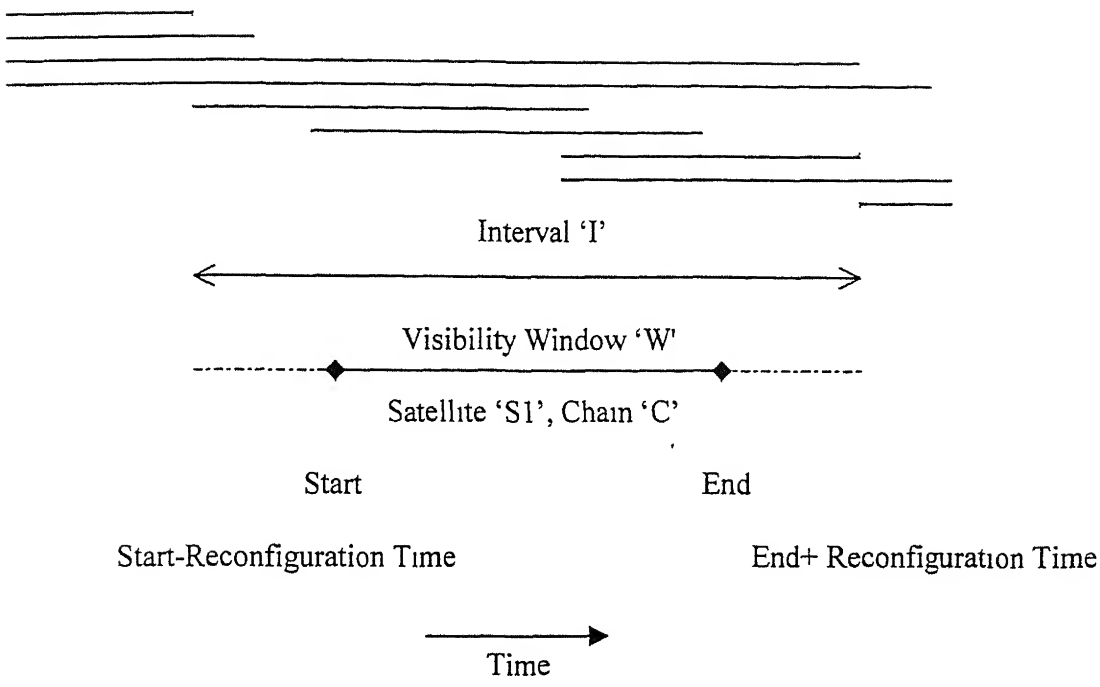
Propagating Controller Clash Constraint: Suppose, W is scheduled to be supported by the controller. This implies that the controller is not available for time interval I. This implication is imposed on all visibility windows of S2 that fall under any of the following cases

A 'AOS' of S2 $\in$ (-∞, start-Preparation Time] AND 'LOS' of S2 $\in$ I
Propagation Of Controller Clash Constraint Is Done As Overlapping part of the visibility window of S2 can no more be used to schedule any operation of S2

B 'AOS' of S2 $\in$ I AND 'LOS' of S2 $\in$ [end + Preparation Time, +∞)
Propagation Of Controller Clash Constraint Is Done As: Overlapping part of the visibility window of S2 can no more be used to schedule any operation of S2.

C. 'AOS' of S2 $\in$ I AND 'LOS' of S2 $\in$ I
Propagation Of Controller Clash Constraint Is Done As: Entire window of S2 can no more be used to schedule any operation of S2.

Figure 3.5.2: Visibility Clash Scenario:



Visibility Clash Scenario 'W' is a visibility window of satellite S1 at a chain 'C'. If any of the visibility windows (of any of the satellites, other than S1) on chain 'C' matches with any of the visibility windows shown as light lines in the figure above, then status of window 'W' is acknowledged as a clashing window (visibility clash)

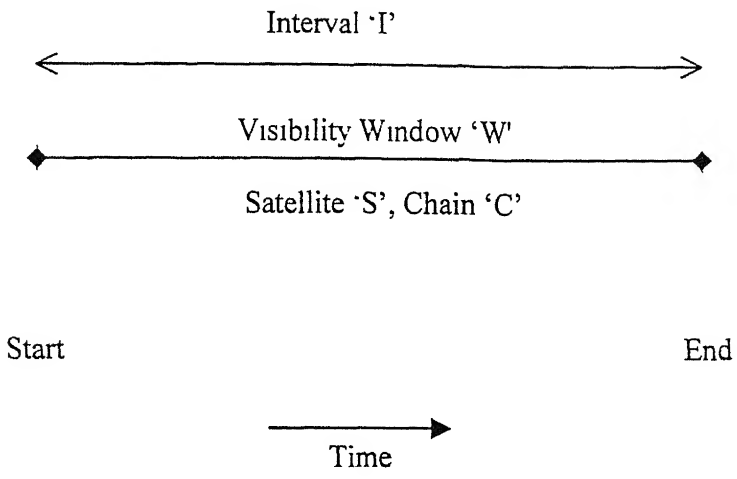
Propagating Visibility Clash Constraint Suppose, W is scheduled to be supported This implies that the chain C is not available to support any other satellite for time interval I. This implication is imposed on all other visibility windows (say W') on chain C that fall under any of the following cases.

B 'AOS' of W' ∈ (-∞, start- Reconfiguration Time] AND 'LOS' of W' ∈ I
Propagation Of Visibility Clash Constraint Is Done As: Overlapping part of the visibility window W' can no more be used to schedule any operation.

B 'AOS' of W' ∈ I AND 'LOS' of W' ∈ [end + Reconfiguration Time, +∞)
Propagation Of Visibility Clash Constraint Is Done As: Overlapping part of the visibility window W' can no more be used to schedule any operation.

C. 'AOS' of W' ∈ I AND 'LOS' of W' ∈ I
Propagation Of Visibility Clash Constraint Is Done As: Entire window W' can no more be used to schedule any operation.

Figure 3.5.3: Inter-Station Clash Scenario:



Inter-Station Clash Scenario 'W' is a visibility window of satellite S at a chain 'C' If at the same time, any part of the visibility window 'W' is visible from a chain other than 'C', then status of window 'W' is acknowledged as a clashing window (inter-station clash).

Propagating *Inter-Station* ***Constraint:*** Suppose, W is scheduled to be supported. This implies that for interval I the satellite S is not available to be supported by any chain other than C This implication is imposed on all other visibility windows (say W') of S on chains other than C that fall under any of the following cases

C 'AOS' of W' $\in$ (-∞, Start] AND 'LOS' of W' $\in$ I
Propagation Of *Inter-Station* ***Constraint Is Done As***: Overlapping part of the visibility window W' can no more be used to schedule any operation of S.

B 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ [End, + ∞)
Propagation Of *Inter-Station* ***Constraint Is Done As.*** Overlapping part of the visibility window W' can no more be used to schedule any operation of S.

C 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ I
Propagation Of *Inter-Station* ***Constraint Is Done As*** Entire window W' can no more be used to schedule any operation of S

Propagating 'precedence' constraint: A precedence constraint on an operation states that some other operation must take place before (or after) it Propagation of minimum duration constraint converts status of some operations from "possible" to "not possible". In turn, status of operations succeeding those whose status have been converted from "possible" to "not possible", also gets converted from "possible" to "not possible"

Step 5 is repeated until no more entity, that has some opportunity to get scheduled, is left. The block diagram 3 6, on next page, summarizes the total approach.

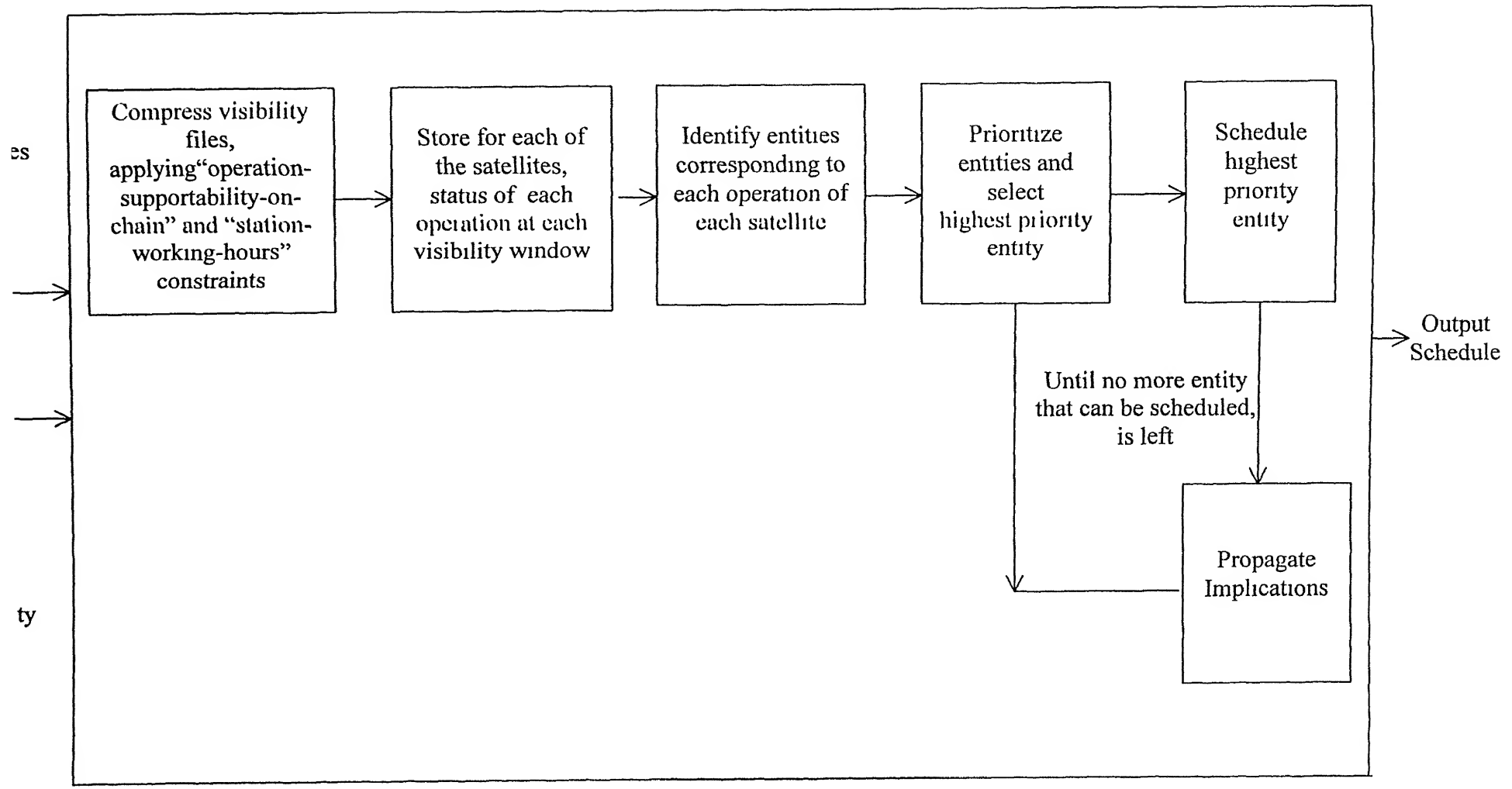


Figure 3.6: Method Block Diagram

In previous chapter we defined *slot* for an operation of a satellite as a set of visibility windows, one out of which demands to be scheduled for this operation of the satellite and combinations of satellite-operation-slot as *entities* We also discussed that number of slots in a reference time is determined by cycle of the operation and there may be many slots in the reference time interval (one week for present problem) for an operation of a satellite. We described how the search process advances and arrives at the final schedule. The approach involves "selecting one entity at a time", the entity being chosen is the one with highest priority value among all contending entities In this chapter we discuss how priorities are assigned to various entities.

Priority value of an entity:

All the satellites move in their fixed orbits and are visible on chains of ground network. It often happens that visibility windows of different satellites occur on a chain simultaneously. Orbit dynamics of the satellites is such that if visibility clash occurs at some chain once, it continues to occur at next many chains and this situation persists for many continuous orbits

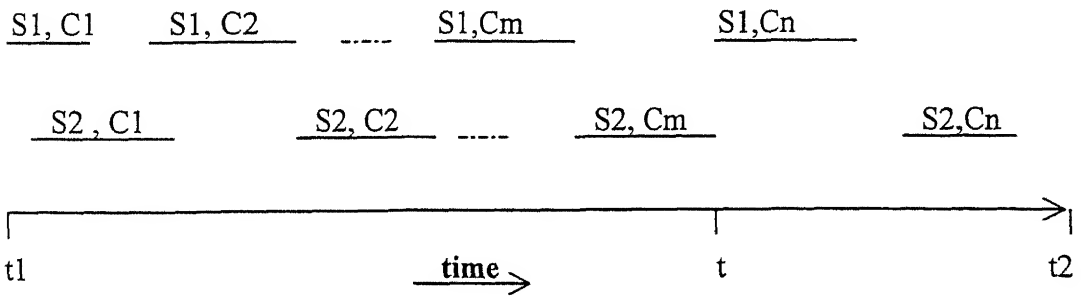


Figure 4.1: Two Continuously Clashing Satellites

Figure 4 1 shows visibilities for two satellites S1 and S2 in time interval [t1, t2].The lines shown above the time-axis are visibility windows of satellites S1 and S2 on chains C1, C2, ...C_n, respectively The first visibility clash of satellites S1 and S2 occurs at chain C1 and then continues to occur at successive chains till time t. We describe now how the optimal resolution of above clash scenario requires considerations of many factors We assume that the priority of supporting S1 is higher than that of S2

- Satellites have relative priorities depending upon the "phase" of satellite's mission – the number of orbits the satellite has completed since launch On considering satellite priority as the deciding factor, all the chains C1, C2, C3. . should support S1 (its priority is higher than that of S2) But this may result in violation of the constraint that gap of more than three orbits between two consecutive supports on a satellite is not permitted. That is, gap between two consecutive TM operations should not be more than three orbits.

- Let us take a clash scenario as shown in Figure 4.2. Two visibility windows of different satellites S1 and S2 are occurring on chain C. Here again, priority of S1 is assumed to be higher than that of S2. The chain can support only one of these satellites. Operations that can be performed in the window of S1 are TM, TC, PB and those that can be performed on the window of S2 are TM, TC. RawSS. Now, RawSS is an important operation Satellite priority of S1 being higher, hence the visibility window of S1 demands for support, whereas importance of RawSS demands support of S2 in the visibility window of S2 Thus we have a conflict here that must be resolved.

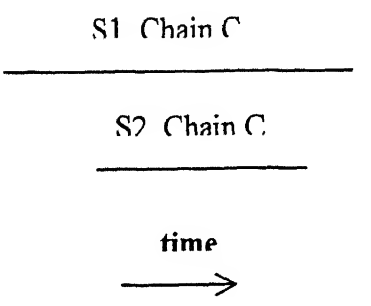


Figure 4.2: Visibility Clash of Two Satellites on a Chain

- There are a number of important operations specific to each of the satellites. All of these require to be done in their slots (slots are sets of visibility windows, such that, one window out of each of these sets requires to be scheduled) and therefore carry equal importance Each of these operations has its own set of windows on which it is possible to perform the operation (operation-supportability-on-chain constraint, minimum duration of operation constraint, etc) These windows for different operations of satellites in different slots vary in number.
- During scheduling, if an entity is missed once or more than once in its immediately preceding slots, the priority of entity's operation in its slot must rise. i e it would become more important to schedule the operation in its slot. And, the larger the number of slots missed, the higher should be the increase in the entity's priority
- We discussed in Chapter 3 that various types of clashes may exist among visibility windows of satellites – controller clash, visibility clash and inter-station clash. These clashes can be very frequent and large in number Because of these clashes, the allocation of a single window for some operation, causes parts of various other windows to get blocked That is, the time available for scheduling at other windows becomes shorter. And, due to "minimum duration for operation" constraint and "precedence" constraint, the status of many operations at the blocked windows becomes "not possible to perform". Also, a satellite may have a set of operations that are performed using same carrier frequency signals So, if an operation is performed on the satellite, which belongs to this set of operations, it restricts performing other operations of the set.

In other words, scheduling an operation of a satellite at a window grabs away opportunities of many other operations of the same and other satellites.

To capture all above factors, we use a priority function defined as follows: -

$$\text{Priority Value (entity}_I) = (k1 / (\text{number of options in the slot of entity}_I + 2)) + k2 + n^* D$$

Above priority function is used to calculate priorities of competing (possible to schedule and not yet scheduled) entities

- The "2" in the denominator takes care that denominator never becomes zero
  We choose values of k1 and k2 arbitrarily as 100 and 150, respectively.
- Number of options in a slot is the number of windows in the slot where status of entity's operation is "possible to perform". 'n' represents number of times scheduling of an entity has been missed in immediately preceding slots and 'D' is increase in priority value of an entity per immediately previous missed slot.

Since scheduling an operation of a satellite at a window, grabs away opportunities of many other operations of the same as well as of other satellites, the entity should be scheduled first that has lesser number of options for its operation to get scheduled. This is taken care of by the first feature (term) of the priority value function. It is a decreasing function with respect to slack. For entities with high slack (these have a large number of options to be scheduled). value of first part of the function is lower than that of entities with low slack (these have very few ways to be scheduled).

- We choose the value of 'D' such that if an entity is altogether missed to schedule (no opportunities in its slot) in a slot, its priority value becomes higher than any other entity's priority value due to first part of the priority value function. And the higher is the number of immediately preceding slots missed, higher should be this increase. The last part of the priority function takes care of this factor. If 'n' number of slots have been missed, the priority function raises this entity's priority by 'n*D',
- In case of tie (two entities are having same priority value), entity of higher priority satellite is selected for scheduling.

Once an entity, with highest priority, is selected and scheduled, it is thrown out of competition and the propagation of implications of this "dropping" is done as discussed in the previous chapter Priority values of the remaining contending entities get altered on propagating these implications.

Thus, priorities of entities change *dynamically* with scheduling of every new entity and with propagating implications.

In our problem, PB and TR operations have been specified as operations with the cycle "once every orbit". However, skipping these operations in an orbit in some orbits is tolerable There is a reason behind why skipping PB in an orbit is allowed

PB operation utilises on-board memory storage on the spacecraft. This memory device monitors health parameters of various subsystems of a satellite and is an example of type of resource that is required during the execution of the task (leading end of PB operation, discussed in Chapter 2) and continues to be required past the completion of the task (until the next PB operation is performed on the satellite). Capacity of the memory device is fixed - it is designed to store just 1 orbit's data. If the device does monitoring in the "sample mode", it can store data for time more than it can store in continuous mode. Thus, this device may be commanded to perform in sample mode– if it has to store more than 1 orbits data

However, we desire PB and TR to be performed maximum times its possible.

Thus, the total set 'S' of operations on satellites that are to be performed on TTC ground stations can be considered to be comprising of two subsets: -

- Set S1. PB and TR, each to be performed very frequently
  (once every orbit)
- Set S2: other periodic operations of satellites, i e excluding those in S.

Here, $S1 \cup S2 = S$

Skipping operations of set S1 is tolerable. Therefore, priority values of entities whose operation is TM/PB and cycle is 'once every orbit' (for some satellites periodicities TM and PB are once/twice per day) may be kept lower (first part as well as last part of priority function) than that of any entity of set S2, although both are having the same number of options in their slots or have been delayed by the same number of slots.

In Chapter 3 and 4, we focused on the method proposed by us to schedule the telemetry, tracking and commanding operations of satellites The amount of information involved to arrive at the final schedule is quite large. A great deal of it consists of the various constraints that are applicable in the task allocation problem In this chapter, we discuss how we have organized these data in various data files We also discuss the programs – the imposition of constraints, function of each program, input files and its output. These programs are coded in C++, which are executed in serial order to arrive at the final schedule

5 1 **Input Data Files:** All the general information about satellites, chains, stations, operations has been organized in *six* data files Table 5 1 summarizes the fields of each of the data files The files are provided in appendix. Details of these data files is as follows. -

File# 1: Total number of records in this file is equal to the number of satellites whose operations are to be scheduled. This file summarizes details of operations, the constraints implied in the execution of these operations, satellite priority number, and also human-controller details for each of the satellites. Details of operations include names of all the operations that are to be done on the satellites, names of operations that are performed on a satellete via carrier signals of same frequency and hence more than one operation from these cannot be performed at same opportunity window, minimum duration in minutes required to perform each of the operations and period of each of these operations. Human controller details include names of satellites controlled by the same controller and time taken by controller to switch support from one satellite to another.

For each record of this file : -

- First field contains name of a satellite;
- Second field contains names of operations performed from ground station on a satellite, using same carrier frequency signals.

If no such set of operations exists, second field is written as *nullset*;

Table 5.1 : Input Data Files' Details

FILE # AND FILE- NAME	ATTRIBUTES								
	TOTAL NUMBER OF FIELDS	FIELD 1	FIELD 2	FIELD 3	FIELD 4	FIELD 5	FIELD 6	FIELD 7	FIELD 8
1 **!OPDurCy.dat**	8	Name of satellite	Set of "same carrier frequency" operations on the satellite	Names of "all" operations on the satellite	Minimum time, in minutes, required to perform each of the operations	Cycles of each of the operations	Satellite priority number	Name of other satellite controlled by the human controller of satellite mentioned in field 1	Controller-preparation time
2 **ImplyOps.dat**	3	Name of satellite	Name of a satellite operation	Names of preceding operations	-	-	-	-	-
3 **TTCStnHr.dat**	8	Name of TTC station	Name of the chains of the TTC station	Shift 1 work hours		Shift 2 work hours		Shift 3 work hours	
4 **CycleDet.dat**	6	Name of satellite and names of	Cycle is expressed as fixed date or	Date when the 'once every cycle' type	Cycle in days	Date when the 'once every	Time (day/night) when the operation	-	-

		operations of type 'once every cycle'	fixed interval (FD/FI)	operation was last performed		cycle' type operation will be next performed	is to be performed =>m/n/x		
5 **SatChns.dat**	2	Name of satellite	Names of TTC chains supporting the satellite	-	-	-	-	-	-
6 **ChnOpEle.dat**	4	Name of TTC chain	Operations supported by the chain	Minimum elevation angle, in degree, required for each of the operations supportable on the chain	Reconfiguration time of the chain	-	-	-	-

- Third field contains names of all operations that are to be performed on the satellite on TTC stations;
- Fourth field contains minimum durations required to perform each of the operations. The durations are expressed in minutes,
- Fifth field contains cycles of each of the operations of satellite. Various cycles are – *all eligible passes* (AELGP), *once every orbit* (ENO: 1), *not more than three orbits gap between two supports* (ENO. 3), *once every day* (TED· 1. x: x), *once every day-morning pass* (TED· 1: m: x), *once every day-night pass* (TED 1.n·x), daily-*morning- first-pass* (TED 1: m.), *daily-night-last-pass* (TED: 1 n: 1), *once every cycle* (OEC), *twice every day* (TED· 2) and *thrice every day* (TED. 3) Operations of type *once every cycle* are those that need to be done once every 1 week or at intervals of greater than 7 days.
- Sixth field contains satellite priority number defined on 1-10 scale.
- Seventh field contains name of other satellite controlled by the controller of satellite mentioned in field 1 If there is no other satellite controlled by the same controller, this field is written as *NON*
- Eighth field contains time taken, in minutes, by the controller of satellite mentioned in field 1 to switch support to satellite mentioned in field 7. In case field 7 is *NON*, this time is marked as '0'.

File # 2: Operations performed on the satellites are not independent operations, a precedence relationship and thus the implicated constraint exists among many these operations. Almost all the operations are preceded with TM in present problem. This data file captures the precedence relationship existing among various operations of satellites and maintains it using three fields. Information about precedence is maintained as a record in this file as. -

- First field contains name of satellite S.
- Second field of the record contains name of *preceded* operation.
- Third field contains names of *preceding* operations.

File # 3: Different TTC stations of the ground network differ in the timings of work-hours, imposing the associated constraints. Some TTC stations are available for 24 hours service whereas others are available to support the satellites according to their specified shift timings. Information about working hours of the stations is contained in this file in the form of records as -

- First field contains name of TTC station.
- Second field contains names of chains of the TTC station.
- Third field contains start of working hours of the station in time interval:

  [00 00· 00, 08 00· 00]
- Fourth field contains end of working hours of the station in time interval.

  [00 00 00, 08: 00 00]
- Fifth field contains start of working hours of the station in time interval

  [08· 00 00, 16· 00 00]
- Sixth field contains end of working hours of the station in time interval:

  [08. 00· 00, 16: 00 00]
- Seventh field contains start of working hours of the station in time interval

  [16: 00: 00, 24· 00 00]
- Eighth field contains start of working hours of the station in time interval.

  [16: 00 00, 24. 00· 00]

File # 4: A satellite may have a set of operations that are required to be performed at a fixed interval - every week, or every month, or fixed date of month. We call such constrained operations as *cyclic* operations. These operations may demand morning pass, or night passes or any time of the day All this information is stored in this data file as records, which are aggregated satellite-wise For each of the satellites that have at least one cyclic operation, a record is maintained which contains following fields -

- First field contains name of the cyclic operation.
- If the operation is to be performed every seven days or every fixed number of days, second field is marked as *FI* (fixed interval); if it is to be performed every month at some fixed date, second field is written as *FD* (fixed date)
- Third field contains the date when the operation was last performed.

- Fourth field contains cycle of the operation in days.
- Fifth field contains the date when the operation is to be performed next time
- Sixth field specifies whether the operation is to be done at night or morning or any time (n/m/x)

File # 5: One of the capabilities of a chain is that a particular chain can support a specified set of satellites. Therefore, even if a satellite is visible at some chain, the visibility window may be not of any use since the chain cannot support the satellite This data file captures this feature of the chains (the satellite supportability constraint for chain) and contains number of records equal to number of satellites

- First field of a record contains name of a satellite.
- Second field contains names of chains that support the satellite.

File # 6: The capabilities of chains also differ in terms of the particular operations it can support and the minimum elevation required to perform each of these operations on it Further, a chain is constrained by certain minimum time to switch its support from one satellite to another. This time is called the reconfiguration time of the chain. This data is contained in this file in the form of records maintained chain-wise.

- First field contains name of a chain.
- Second field contains names of operations the chain supports
- Third field contains, required minimum elevation angle of visibility windows, to perform various operations supportable by the chain.

5.2 **Visibility Files:** During its motion round the earth, the satellites are visible on various chains of ground network. Each of these visibilities are mentioned as a record in visibility files, prepared separately for individual satellites. Sample of visibility records for January 10, 2000, for the satellite I1A are shown below.

Sample of visibility records for satellite I1A: -

```
2000  1 10 IRS-1A   LK2   60225 11.431   8  3 40   8 15 50
2000  1 10 IRS-1A   SH1   60225 79.678   9 39 12   9 55 28
2000  1 10 IRS-1A   LK2   60225 11.431   8  3 40   8 15 50
2000  1 10 IRS-1A   SH1   60225 79.678   9 39 12   9 55 28
......
```

We can see from above, that a typical visibility record consists of number of fields Each of these fields conveys some specific information about the satellite's visibility

- *Field 1* year
- *Field 2* month
- *Field 3.* date
- *Field 4.* name of satellite
- *Field 5* name of station
- *Field 6.* orbit number
- *Field 7·* elevation angle
- *Field 8,9,10* . AOS (Acquisition Of Signal) hour, AOS minute, AOS second
- *Field 11,12,13* LOS (Loss Of Signal) hour, LOS minute, LOS second

The program we have developed uses weekly visibility files for all satellites and each of the data files explained above, as inputs.

5.3 **Implementation Steps:** The schedule development method proposed, uses *six* data files and the visibility files of all the satellites, (for one week) as its input A number of steps are used to arrive at the final schedule and these steps have been implemented using *ten* programs (C++ code) that are serially linked and are made to run sequentially to arrive at the final schedule. The programs have been developed on the Windows 95 platform. We discuss next, what input files are used by each of these programs, the function of each program and the outputs they produce.

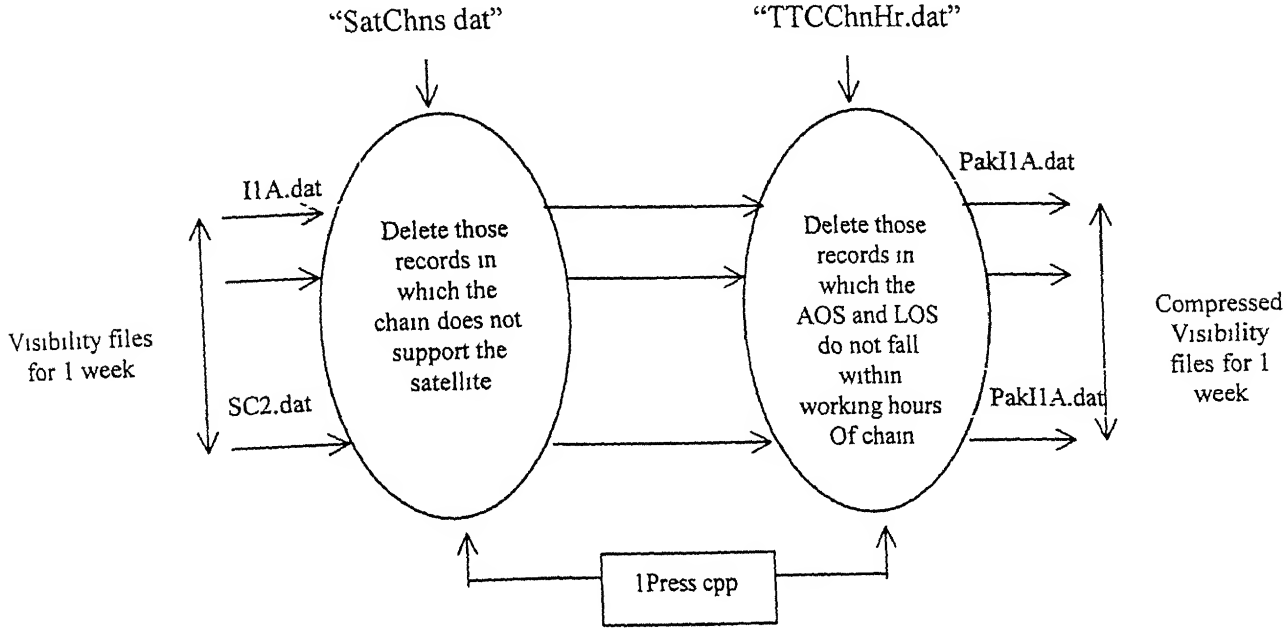
Program #1: "1Press.cpp": ***Compresses each of the visibility files on the basis of satellite-supportability-on chain constraint and station-working-hours constraint.***

Some records in the visibility files do not satisfy "TTC station working hours constraint" or "satellite supportability in chain constraint" These are redundant records, explained as follows: -

- If a satellite is not supportable by a chain, its visibility even if it occurs at this chain, is redundant
- Visibility windows, which do not fall within working hours of a station are also redundant, even if mentioned in the visibility file

The program 1Press.cpp filters out these redundant records. Figure 5 1, explains the features of this program.

Figure 5.1: Features Of 1Press.cpp



Program #2: "2MyFiles.cpp": ***Creates files for internal purpose, corresponding to compressed visibility files (output of program# 1) of each of the satellites.***

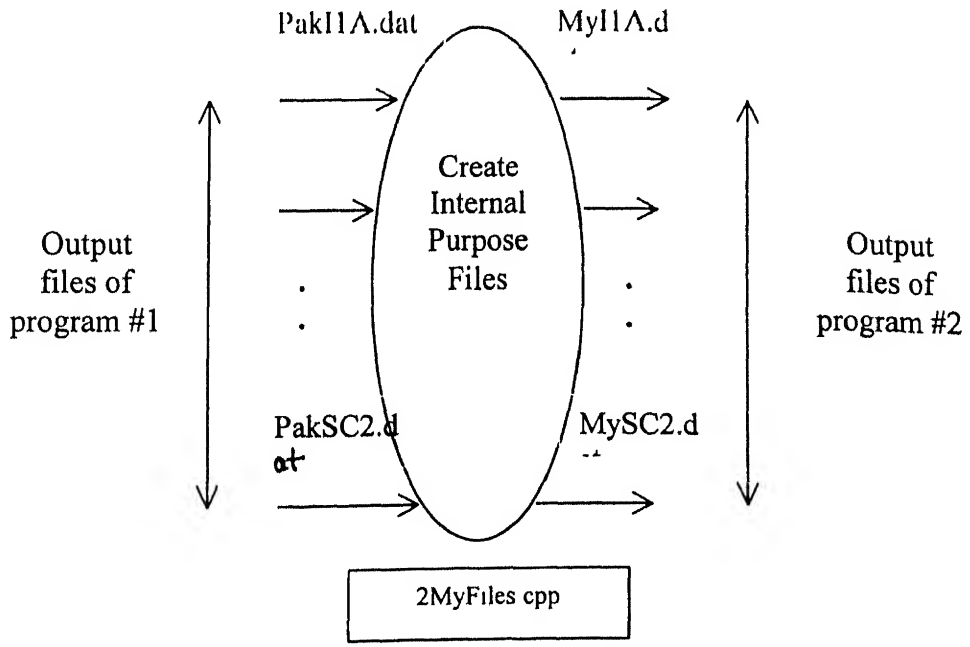
In Chapter 3, we defined *slot* as a set of visibility windows – scheduling a satellite's operation means allocating the operation to one of the visibility windows of this set Slot is determined by the cycle of a satellite's operation. Thus, if the cyclicity of some operation of a satellite is 'once every day', there will be 'seven' slots for this

operation of the satellite in one week. And each of these one-day slots are disjoint sets of visibility windows, containing visibility windows of a particular day of the week. Similarly, for an operation to be performed 'once every orbit', the number of slots in one week would be equal to the number of orbits made by the satellite in one week, and each slot here will consist of visibility windows occurring in the orbit. Also we recall that we termed each of the satellite-operation-slot combination as *an entity*

Entities are basic objects of the problem Total number of entities in present problem is around 1300 – the number is quite large. For identifying each of the entities and maintain status of operations of satellites as 'possible to perform (p)' / 'not possible to perform (n)', at each of the visibility records of the spacecraft, this *program creates a file corresponding to each of the compressed visibility file (output of program# 1). These files are used for internal purpose by the software. The total number of records in each of these internal-purpose files is same as that in corresponding compressed visibility file.*

Figure 5.2, illustrates the function of this process and shows how a compressed visibility file (output of program # 1) is mapped into an internal file (output of program # 2). Example 5.3.2 illustrates the input and output of 2MyFiles.cpp.

Figure 5.2: Features of 2MyFiles.cpp



Example 5.3.2: Sample Files

Input of program # 1(Pakl1A.dat): -

This is also an output file of program # 1.Different field numbers are marked in bold. -

```
  1    2  3   4        5       6      7       8  9 10   11 12  13

2000   1 10 IRS-1A    SH1    60225 79.678    9 39 12    9 55 28
2000   1 10 IRS-1A    LK2    60225 71.673   09 57 04    9 59  5
2000   1 10 IRS-1A    BR1    60228 25.925   13 17 52   13 33  5
2000   1 10 IRS-1A    LK2    60233 67.943   22 21 13   22 29 13
. ..
```

Output of program # 2(Myl1A.dat): -

This file is create by the program for its internal-purpose.
Different field names and numbers are marked in bold: -

```
Elevation      AOS          LOS      Slots Field  Status Field <        -  Priority Values Fields  -              >
   1        2  3  4    5  6  7        8            9          10          11          12          13          14

79.678      9 39 12    9 55 28    !!!!!      ppppp    !!!!!! ! !!!!!! ! !!!!!!.! !!!!!!.! !!!!!!.!
71.673      9 42 48    9 59  5    !!!!!      ppppp    !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! ! !!!!!!.!
25.925     13 17 52   13 33  5    !!!!!      ppppp    !!!!!! ! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! !
67.943     22 21 13   22 37 38    !!!!!      ppppp    !!!!!!.! !!!!!! ! !!!!!!.! !!!!!!.! !!!!!! !
....
```

Field numbers 1-6, of a record number in internal-purpose file of a satellite, are same as field numbers 7,8,9,10,11,12, and 13 of the same record number in compressed file of the satellite For a record number in internal-purpose file of a satellite, year, month, date, and orbit number can be read from the same record number of the corresponding compressed file (output file of program # 1, field numbers 1, 2, 3, and 6) Field number 8 in internal file is used for marking starts and ends of the slots in ith column, slots for ith operation are marked Field number 9 stores status of each of the five operations of I1A· in ith column, status for operation # 'i' is marked The status for all five operations of I1A in field number 9 is initially marked as 'p' Field number 10 onwards are used for storing priority values of entities, 10^{th} field stores priority values for entities of operation # 1, 11^{th} field stores priority values for entities of operation # 2, and so on

Before describing the use of other fields (field number 7 onwards) of internal-purpose files, let us map each of the operations of a satellite to an integer value, this would make our further discussions, easy to understand· -

We follow the sequence of operations of a satellite as in field number 3 of the input data file "InOpDurCy dat", and assign an integer number to each of these, such that first operation is assigned the number '1', second operation is assigned number '2', and so on

In this way, for satellite I1A, the operations TM, TC, TR, PB, VHF_TC get numbered as 1, 2, 3, 4 and 5 respectively and we now call these operations as operation # 1, operation # 2, ... We apply this convention for all operations required for each of the satellites. Next, we explain what field numbers, eight onwards, are meant for in the inter-purpose file of a satellite. Further details of the internal-purpose files are as folows· -

Field number 8 of the internal-purpose file of a satellite maintains starts of slots (marks as *) and ends of slots (marks as ~) information, for each of the operations of the satellite. This field contains a number of columns equal to the total number of operations on the satellite. First column maintains slots' starts and ends information for operation # 1 of the satellite, second column maintains slots' starts and ends information for operation # 2 of the satellite, . . and so on The last column maintains slots' starts and ends information for last operation the satellite.

Field number 9 of internal-purpose file of a satellite also contains the number of columns equal to number of operations on the satellite, first column maintains status of operation # 1 of the satellite, second column maintains status of operation # 2 of the satellite, .. and last column maintains status of last operation number of the satellite. In Figure 5.3.2, in each of the visibility records in "MyI1A.dat", the status of all operations is 'p'; meaning that all the operations of the satellite are possible to be performed on each of these visibility windows. This 'p' entry is made in an initialization step.

Field number 10 onwards maintains priority values of entities – field number 10 maintains prioritiy values of entities for operation # 1 of the satellite, field number 11 maintains prioritiy values of entities for operation # 2 of the satellite, .. and so on.

The symbol '!' represents that no marking of slots has yet been done (field 8) or no priority value has yet been stored (field 10 onwards).

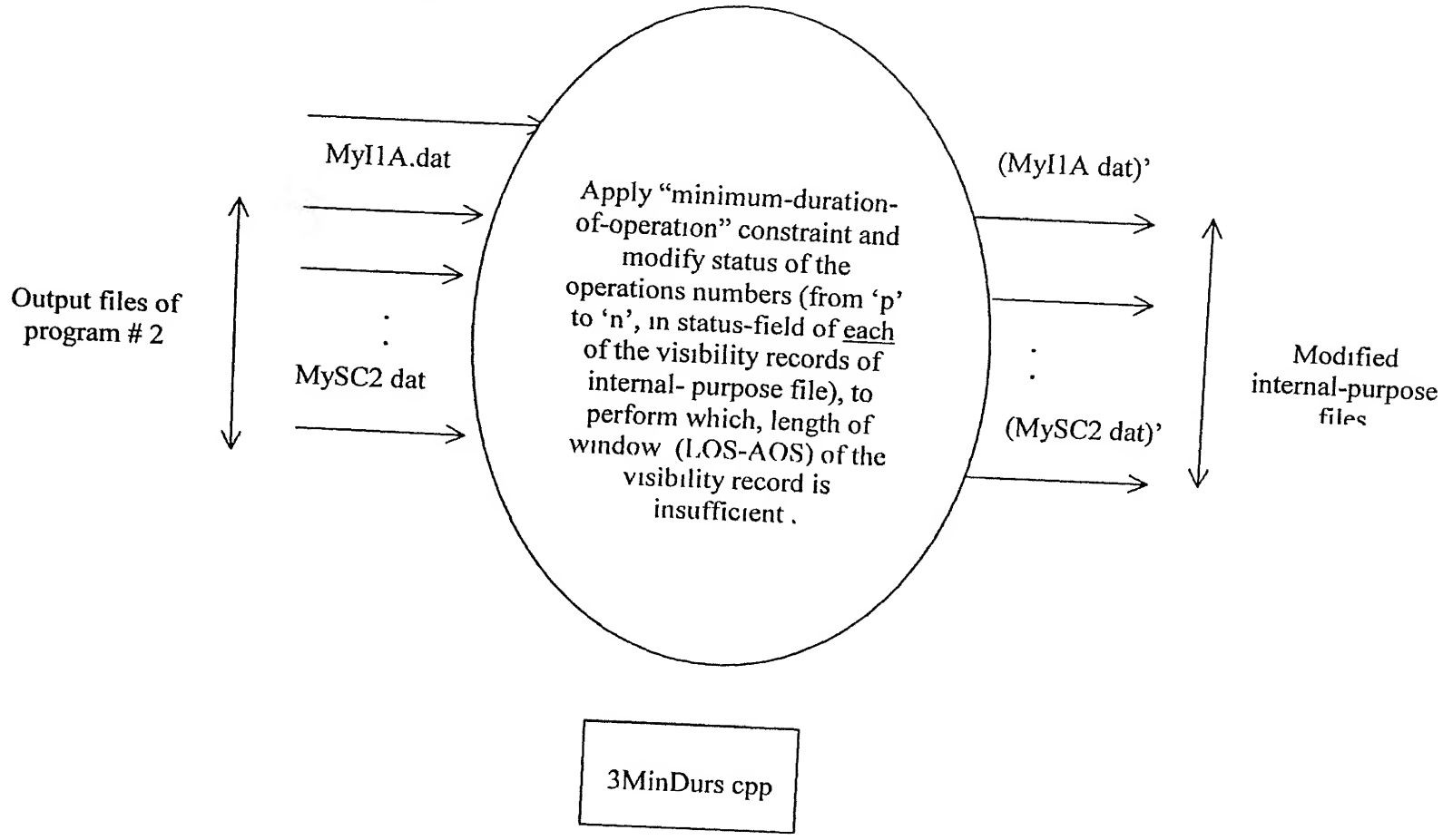
Program #3: "3MinDurs.cpp": ***Applies "minimum-duration-of-operation"constraint on each of the visibility records of all internal files. Thus it maintains the status of all operations (in status-field of internal-purpose file) of a satellite as ' p (possible to perform)' / 'n (not possible to perform)', for each of the visibility record being displayed.***

In the internal-purpose files of each of the satellites, field number 8 of each of the record numbers is used to store status of operations of satellite as ' p (possible to perform)' / 'n (not possible to perform)' These status are initialized as 'p' in all the internal-purpose-files, for all operations at each of the records, by program # 2

Each satellite support operation requires a minimum duration to be performed. Program # 3 traverses each of the internal-files record wise, and at each of the record, calculates the difference of AOS (expressed in field numbers 5, 6, 7) and LOS (expressed in field numbers 1, 2. 3). Then it compares this value to minimum duration required to perform each of the operation numbers of the satellite And, if for an operation number, this difference is less than the minimum duration required to perform the operation, status of this operation number is written as 'n (not possible to perform)' in status field's column number equal to operation number.

Figure 5.3.3, on next page, shows the function of this program and also the input files used by the program. In the example 5.3 3, we illustrate a sample input and output It is observable from this example that some of the status of operations have got converted from 'p' to 'n' due to insufficient window length.

Figure 5.3: Features of 3MinDurs.cpp



Example 5.3.3: Sample Files

Input of program # 3:

This is a sample of internal file

```
4.275       0   3  46    0  11  49   !!!!!!!   ppppppp   !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!! !  !!!!!!.!
!!!!!!.!
79.993      1  39  14    1  54  18   !!!!!!!   ppppppp   !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!! !
 1.583      3  23  11    3  28  19   !!!!!!!   ppppppp   !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!!.!
32.000     4  53  26     5   7  34   !!!!!!!   ppppppp   !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!!.!
.....
```

Output of program # 3:

In the internal file the status is modified on imposing "minimum duration of operation" constraint

```
4.275       0   3  46    0  11  49   !!!!!!!   ppppppp   !!!!!! !  !!!!!!.!  !!!!!!.!  !!!!!! !  !!!!!!.!  !!!!!!.!
!!!!!!.!
79.993      1  39  14    1  54  18   !!!!!!!   ppppppp   !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!!.!
 1 583      3  23  11    3  28  19   !!!!!!!   nppppppp  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!!.!
32.000     4  53  26     5   7  34   !!!!!!!!  ppppppp   !!!!!!.!  !!!!!!.!  !!!!!! !  !!!!!!.!  !!!!!!.!  !!!!!!.!
!!!!!!.!
.....
```

modified status: p -->n (due to insufficient window length)

Program #4: "4OSOC.cpp" ***Applies "operation-supportability-on-chain" constraint and "minimum-elevation-angle-to-perform-an-operation-on-a-chain" constraint on each of the visibility records of all internal files, thus modifies (from 'p' to 'n') status of those operation numbers (in status-field of internal-purpose file, for each of the visibility records of a satellite) that are not satisfying these constraints.***

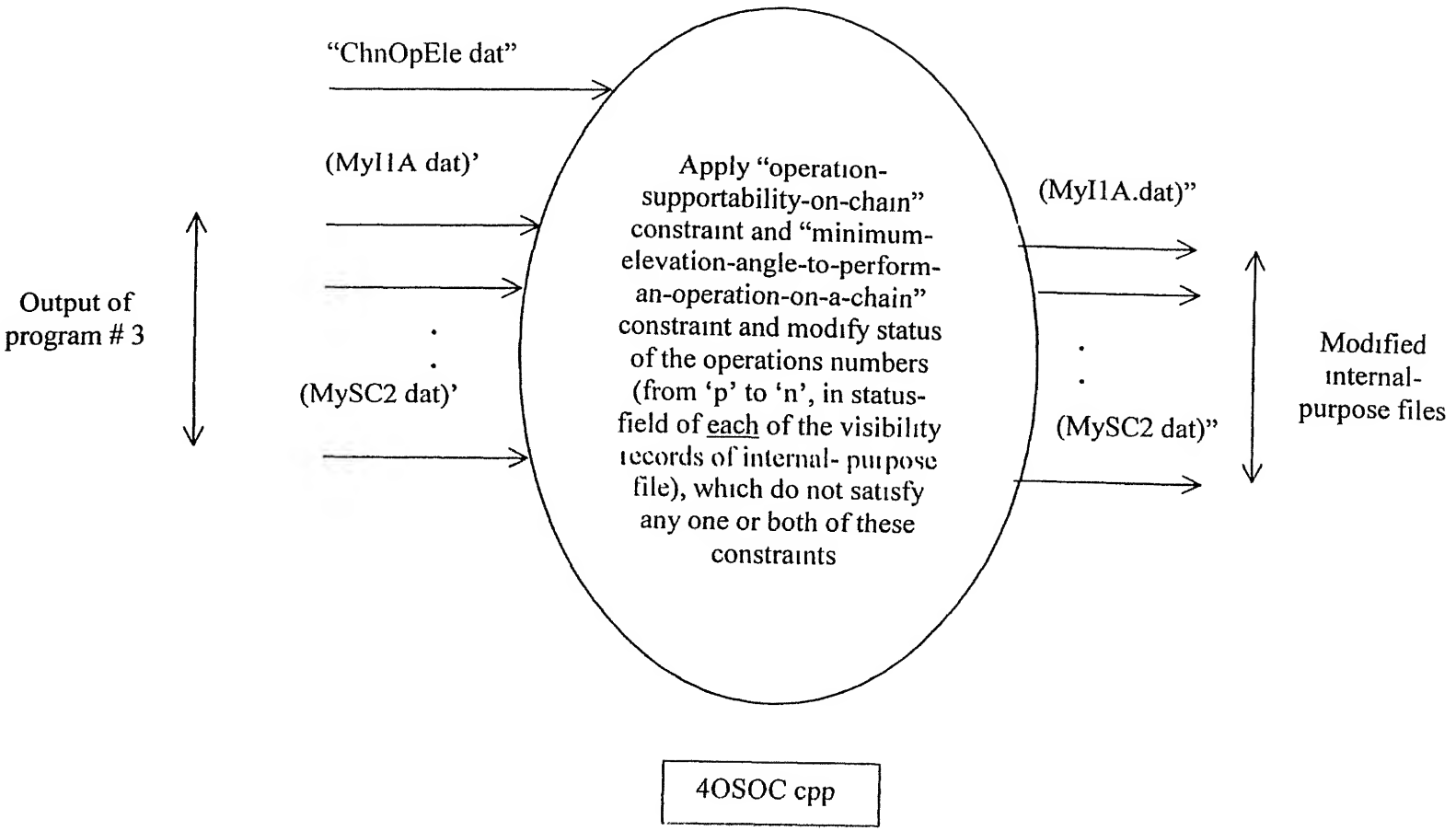
This program forces "operation-supportability-on-chain" constraint and "minimum-elevation-angle-to-perform-an-operation-on-a-chain" constraint and thus further modifies the status of operations in each of the internal-purpose file and at each of the records If an operation number at a record of an internal file, is not supportable by the chain in that
record, its status is maintained as 'n (not possible to perform)' Or, if the elevation angle (field number 1 of internal purpose file), at a record of an internal file is less than the minimum elevation angle required to perform an operation number, then also, status of this operation number is maintained as 'n (not possible to perform)' in this record of internal file.

Figure 5.4, on next page, shows the function performed by this program and also the input files used by the program.

Program #5: "5SetOps.cpp" ***Applies "precedence-relationship-among-operations" constraint on each of the visibility records of all internal files, thus modifies status (from 'p' to 'n') of those operation numbers (in status-field of internal-purpose file, for each of the visibility records of a satellite) that are not satisfying this constraint.***

Precedence relationship exists among many operations of a satellite. Therefore, it is not possible to perform an operation in a window of a visibility record, if status of any of operations that should precede the operation is 'n (not possible to perform)' at that window. This program imposes the precedence constraints on each of the operations of a satellite at the visibility records level – the status

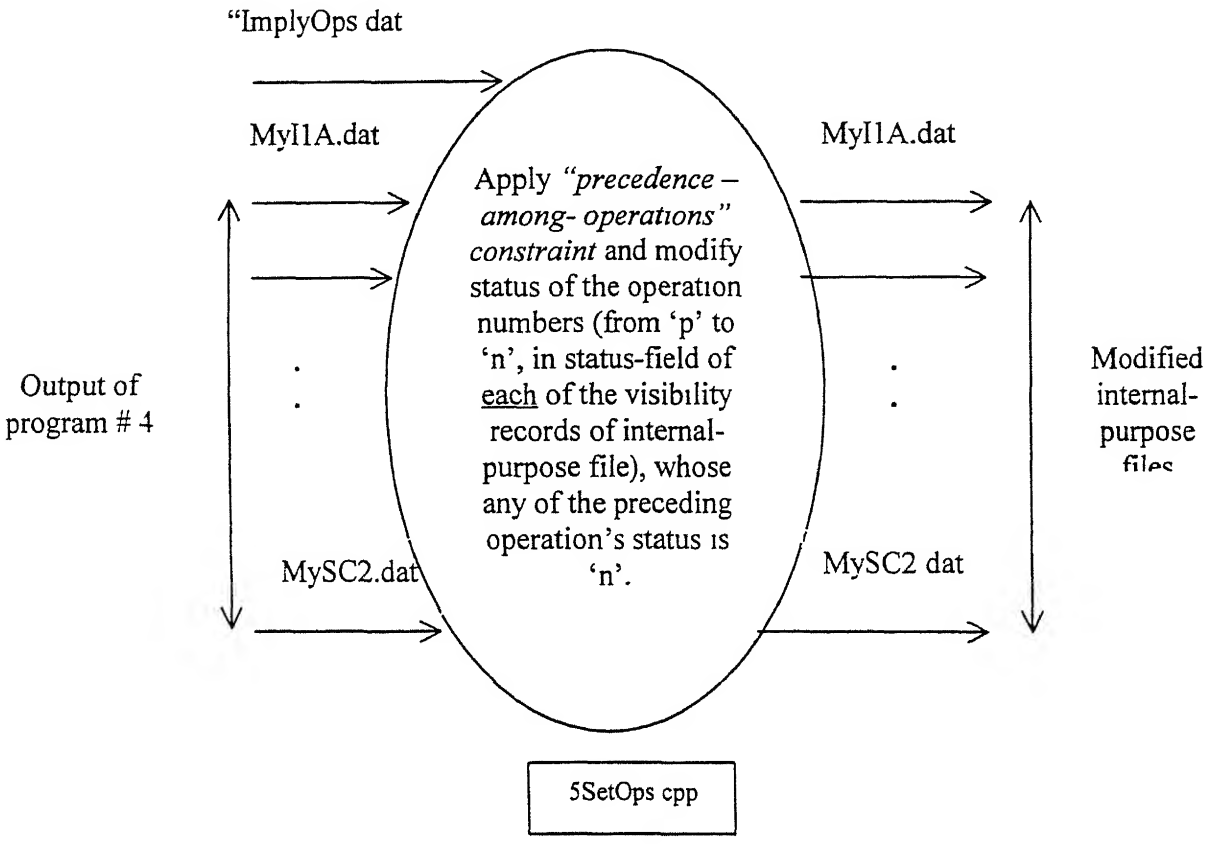
Figure 5.4: Features of 4OSOC.cpp



of any of the preceding operations of an operation number, say 'i' is 'n (not possible to perform)', the program converts the status of this operation number 'i' from 'p (possible to perform)' to 'n (not possible to perform)'.

To complete is function, the program takes precedence relationship data from the data file "ImplyOps dat" Figure 5.5 summarizes the features of this program In example we show a sample of records in which the status of operations changes due to a precedence constraint existing -

Figure 5.5: Features of 5SetOps.cpp



Example 5.3.5: Sample Files

Input of program # 5: -

This is a sample internal file

```
9.986  10 33 55   10 43   5    !!!!!!  pppnnn  !!!!!!.! !!!!!! ! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
 9.985  10 33 55   10 43   5   !!!!!!  pppnnn  !!!!!! ! !!!!!! ! !!!!!!.! !!!!!! ! !!!!!! ! !!!!!!.!
 1.516  10 40  1   10 44 18    !!!!!!  npppppn !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
44.164  12 11  0   12 22 19    !!!!!!  pppppp  !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! ! !!!!!! !
```

Output of program # 5: -

```
9.986  10 33 55   10 43   5    !!!!!!  pppnnn  !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
 9.985  10 33 55   10 43   5   !!!!!!  pppnnn  !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
 1.516  10 40  1   10 44 18    !!!!!!  nnnnnn  !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! !
44.164  12 11  0   12 22 19    !!!!!!  pppppp  !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
```

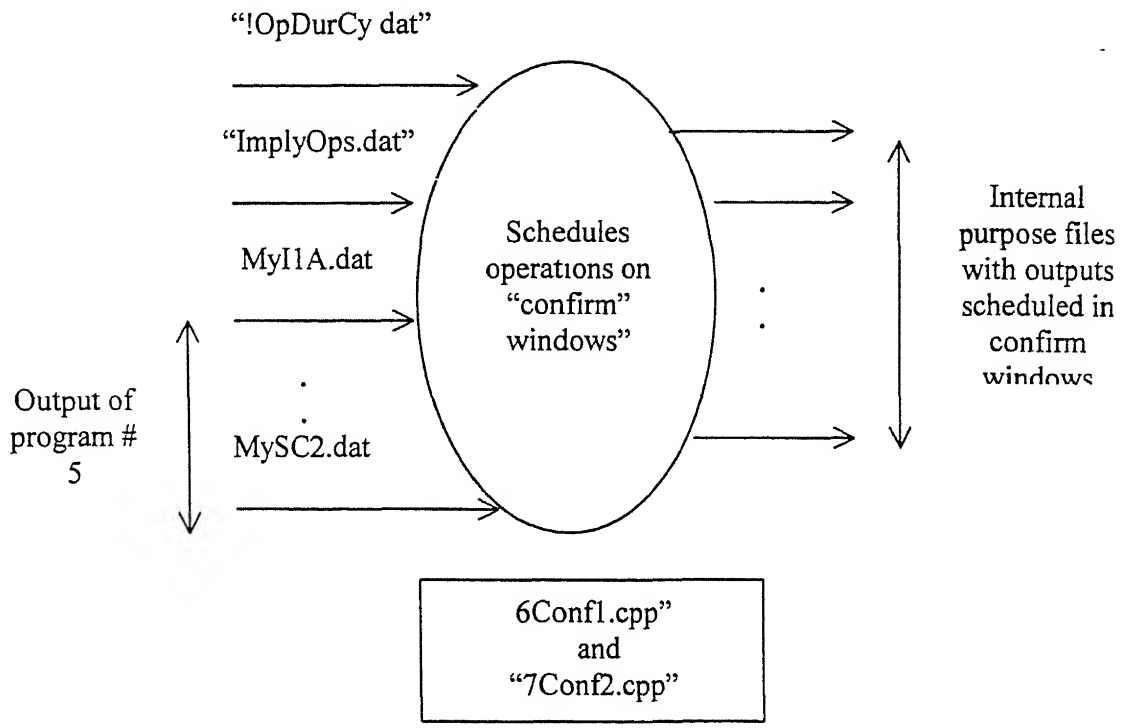
status of all operations has got modified to 'n'

In the output of program number 5, the status of all operations in third record has got converted from 'n' to 'p'. This is because the status of the first operation is 'n' in input file at this record number and because the first operation should precede every other operation numbers, from 2 to 6

Program # 6 and 7: "6Conf1.cpp" and "7Conf2.cpp" : *Schedule operations in "no-clash" windows*

We discussed "no-clash" windows, also called as "confirm windows" in Chapter 3 Scheduling an operation in such window does not block any part of any the other visibility windows These programs identify all "no-clash" visibility records for each satellite and then schedule the operations that are in "possible- to-perform (p)" status at these records. Figure 5.6 explains the inputs taken by the programs and corresponding output. Example 5 3 6 shows the status of scheduled operations at a window as 's'

Figure 5.6:Features of 6Conf1.cpp and 7Conf2.cpp



Example 5.3.6:Sample Files

Input :

This is an output file (internal file) of program number 5

```
 ..
63.616   9 47 52   10  4  7   !!!!!   ppppp   !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! !
24.075  13 23  4   13 38  8   !!!!!   ppppp   !!!!!!.! !!!!!!.! !!!!!! ! !!!!!!.! !!!!!!.!
60.429  22 26 18   22 42 37   !!!!!   ppppp   !!!!!!.! !!!!!!.! !!!!!!.! !!!!!! ! !!!!!!.!
14.883  8 13 9     8 26 17    !!!!!   ppppp   !!!!!!.! !!!!!! ! !!!!!!.! !!!!!! ! !!!!!! !
```

..

Output:

```
 ..
63.616   9 47 52   10  4  7   !!!!!   ppppp   !!!!!!.! !!!!!! ! !!!!!!.! !!!!!!.! !!!!!!.!
24.075  13 23  4   13 38  8   !!!!!   ppppp   !!!!!! ! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
60.429  22 26 18   22 42 37   !!!!!   sssss   !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
14.883  8 13 9     8 26 17 !!!!!     sssss   !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.! !!!!!!.!
```

.....

status of operations has got changed from 'p' to 's' at "confirm windows"

The above example shows that the status of all operations at some records has changed to 's' ; the program identifies these windows as "confirm windows" and schedules the operations that are possible to perform in these windows

Program # 8: "8Slots.cpp": ***Marks slots for all the operations of each of the satellites in internal-purpose files.***

For an operation of a satellite, slots are determined by the cyclicity required of the operation Each slot is identified by a *start* and *end,* both of these extremes of a slot being expressed in terms of a visibility record number. In field number 8 of internal files, the first column is utilized to mark the extremes of slots (start and end) for operation # 1, second column is utilized to mark the extremes of slots (start and end) for operation # 2, .. and so on. It is program # 8 that does this marking Marked internal file, for satellite I1A is shown below: -

```
11.431   8  3 40   8 15 50  *g**g  ppppp  000000 0 !!!!!!.! 000000.0
000000.0 !!!!!!.!
79.678   9 39 12   9 55 28  !g!!g  ppppp  !!!!!!.! !!!!!!.! !!!!!!.!
!!!!!!.! !!!!!!.!
71.673   9 42 48   9 59  5  !g!!g  ppppp  !!!!!!.! !!!!!!.! !!!!!!.!
!!!!!! ! !!!!!! !
25.925  13 17 52  13 33  5  !g~!g  ppppp  !!!!!!.! !!!!!!.! !!!!!!.!
!!!!!!.! !!!!!!.!
67.943  22 21 13  22 37 38  ~g@~g  ppppp  !!!!!!.! !!!!!!.! 000000.0
!!!!!! ! !!!!!!.!
13.094   8  8 23   8 21  4  *g**g  ppppp  000000.0 !!!!!!.! 000000.0
000000.0 !!!!!!.!
89.554   9 44 14  10  0 31  !g!!g  ppppp  !!!!!!.! !!!!!!.! !!!!!!.!
!!!!!! ! !!!!!!.!
63.616   9 47 52  10  4  7  !g!!g  ppppp  !!!!!! ! !!!!!!.! !!!!!!.!
!!!!!! ! !!!!!!.!
24.075  13 23  4  13 38  8  !g~!g  ppppp  !!!!!!.! !!!!!!.! !!!!!!.!
!!!!!!.! !!!!!!.!
60.429  22 26 18  22 42 37  ~g@~g  sssss  !!!!!!.! !!!!!!.! 000000.0
!!!!!!.! !!!!!!.!
...
```

In this file, the symbols '*' and '~' represent start of a slot and end of a slot, respectively Note also that, each '*' symbol is succeeded by a '~' symbol, the pair represents the start and the end of a slot. In case, there is only <u>one</u> visibility record in a slot, the symbol '@' represents both start and end of the slot. For an operation number say 'i' that is not to be performed in the reference time of one week (date when it is to be scheduled does not fall within the dates of the week and therefore there are no slots for

this operation in the reference week), the program marks 'g' (first letter of 'garbage') in column 'i' in field 8 (slot field) of internal file Corresponding to every */@, say at record number 'r', there is a value written as "000000.0" in the record number 'r' – if we count field number 10 in above internal file as 1, field number 11 as 2, field number 12 as 3, ... and so on, then the field number that maps to number 'i', stores the value "000000.0" as priority value of the operation number 'i' of satellite for the slot starting at record number 'r'. The symbol '!!!!!!.!' is used at a record number 'r' in field number '9+i' when slot for operation number 'i' of the satellite does not start at this record.

To identify these slots, the program reads the cyclicity of the operations from input data file "!OpDurCy dat". To extract information about date (year-month-day) and orbit number for some record number in internal file of a satellite, the program reads the corresponding record number in compressed file (output of program # 1) of the satellite. Figure 5.7. on next page, summarizes the function of this program and the input files it uses

Program #9: "9Sched.cpp": ***Schedules contending entities by iteratively performing a sequence of three actions – prioritize entities and select the highest priority entity, schedule the highest priority entity, and propagate constraints.***

This program schedules the contending entities in the files used meant for internal-purpose of the programs. For this the program imports the information from the data files into linked lists and then uses this information in scheduling the entities To schedule all the competing entities, this program performs a sequence of three steps iteratively until no entities are left in the scenario that have one or more opportunity to get scheduled (Figure 5.8)

In 'prioritization step, priorities of entities are written in the internal files using the priority value function mentioned in Chapter 4 From this prioritized set of entities, the entity with highest priority is selected and scheduled. In case of tie, entity for the highest priority satellite is selected. Having selected the entity, it is scheduled at the first

Figure 5.7: Features of 8Slots.cpp

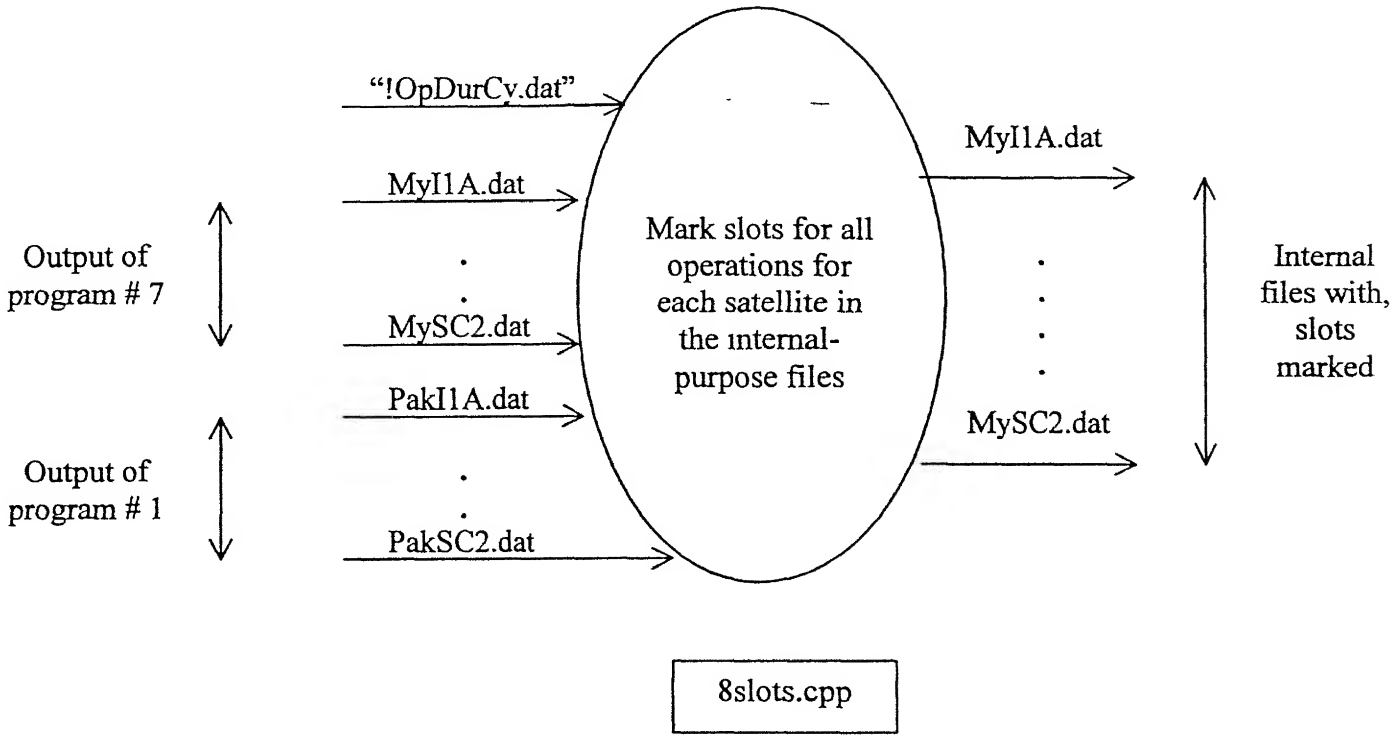
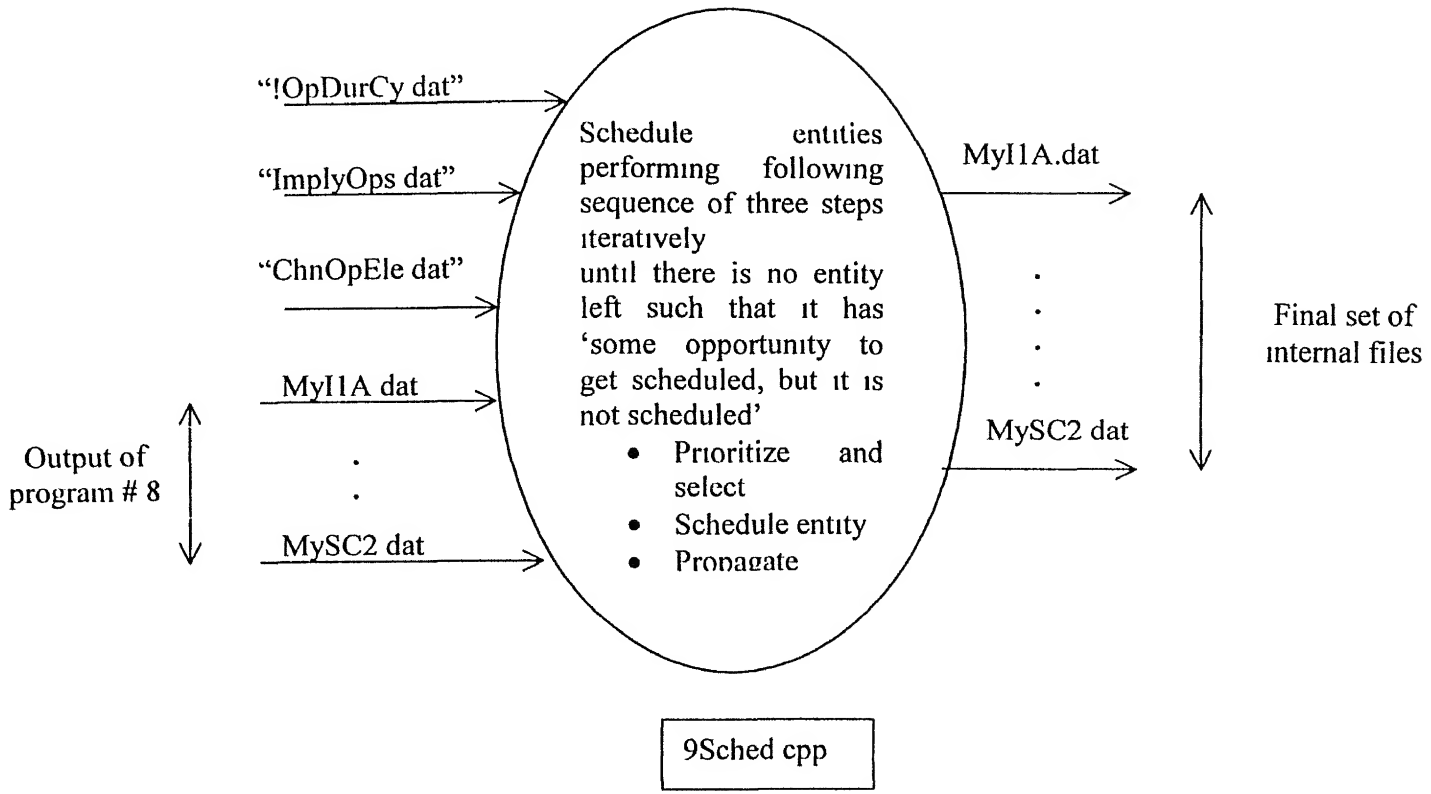


Figure 5.8: Features of 9Sched.cpp



window in its slot and then following six constraints are propagated in the mentioned sequence: -

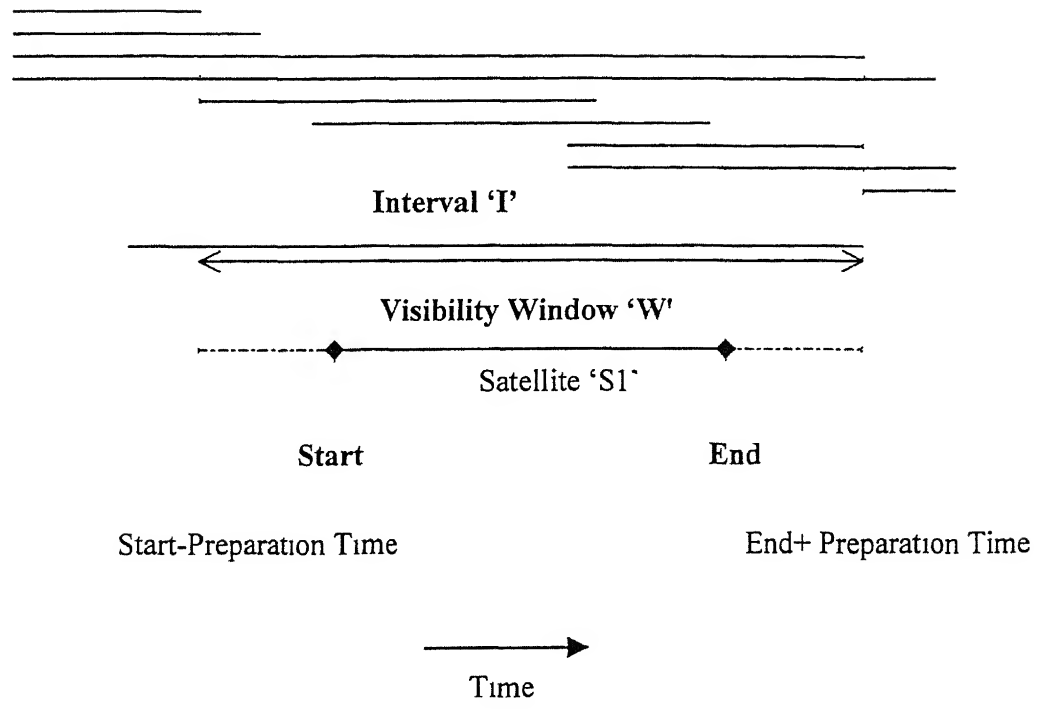
i 'One-out-of-all same carrier frequency operations' constraint
ii 'Controller clash' constraint
iii 'Visibility clash' constraint
iv 'Inter-station clash' constraint
v 'Minimum duration' constraint
vi 'Precedence' constraint

We have discussed propagation of these constraints in Chapter 3. In Figures 5 9, 5 10 and 5 11 we summarize how the propagation of constraints (ii), (iii) and (iv) is done To propagate 'one-out-of-all same carrier frequency operations' constraint, status of all same-carrier-frequency operations other than the one selected for scheduling is changed to 'n' For propagating 'minimum duration' constraint, at each of the visibility windows, status of the operations for which the length of window is insufficient, is converted to 'n'. For propagating precedence constraint, at each of the visibility windows, status of the operations for which any of its preceding operations is in 'n' status, is converted to 'n'

Program #10:"10Format.cpp" ***Generates formatted output schedule for each of the satellites from the corresponding internal-purpose files.***

If an operation number 'i' of a satellite is scheduled at the window of a visibility record, its status is marked as 's' at position 'i' in field number 9 (status field) at this record of internal-purpose file corresponding to the satellite. This program generates the weekly formatted schedule from these internal purpose files. For this, it uses the original visibility files of satellites, compressed (applying "satellite's supportability on chains" and "station-working-hours constraint" on original visibility files) visibility files of the satellites and the internal-purpose files for all satellites (Figure 5.12).

Figure 5.9: Controller Clash Scenario:



Controller Clash Scenario: 'W' is a visibility window of satellite S1 under a controller The same controller has to control another satellite S2. If S2's any of the visibility windows (regardless of the chain on which it is visible) matches with any of the visibility windows shown as light lines in the figure above, then status of window 'W' is acknowledged as a clashing window (controller clash)

Propagating Controller Clash Constraint Suppose, W is scheduled to be supported by the controller This implies that the controller is not available for time interval I. This implication is imposed on all visibility windows of S2 that fall under any of the following cases

A.'AOS' of S2 $\in$ (-∞, start-Preparation Time] AND 'LOS' of S2 $\in$ I

Propagation Of Controller Clash Constraint Is Done As: Convert LOS of this overlapping window to 'Start'.

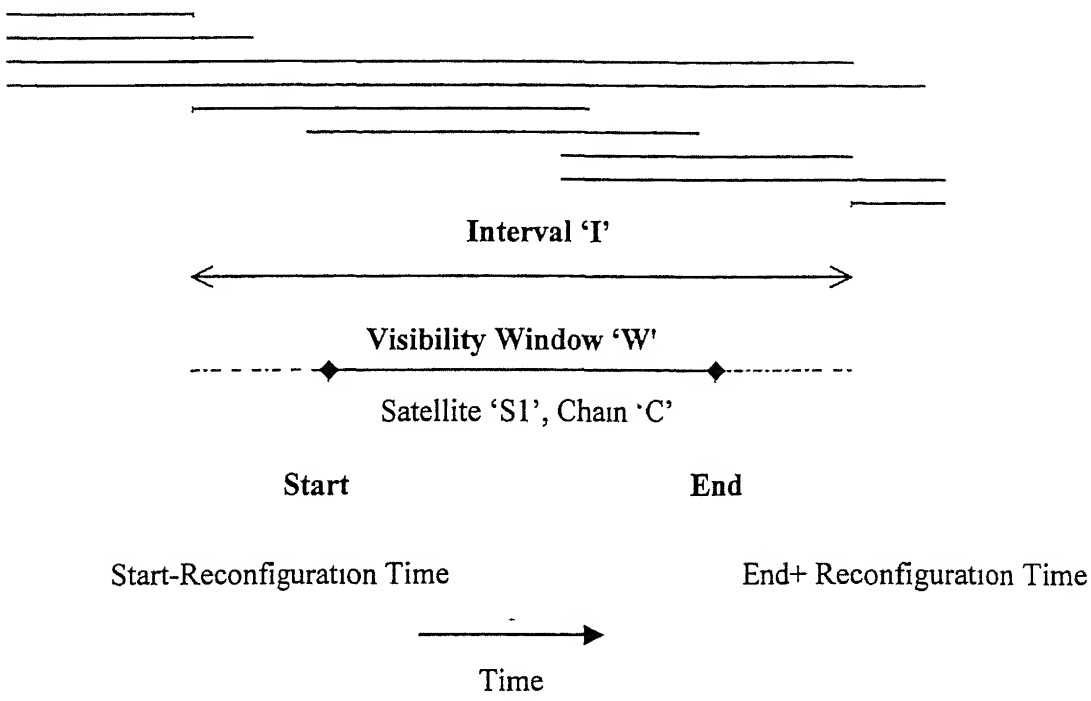
B.'AOS' of S2 $\in$ I AND 'LOS' of S2 $\in$ [end + Preparation Time, +∞)

Propagation Of Controller Clash Constraint Is Done As Change AOS of this overlapping window to 'End'.

C. 'AOS' of S2 $\in$ I AND 'LOS' of S2 $\in$ I

Propagation Of Controller Clash Constraint Is Done As: Change status of all operation in this overlapping window to 'n'.

Figure 5.10: Visibility Clash Scenario:



Visibility Clash Scenario: 'W' is a visibility window of satellite S1 at a chain 'C' If any of the visibility windows (of any of the satellites, other than S1) on chain 'C' matches with any of the visibility windows shown as light lines in the figure above, then status of window 'W' is acknowledged as a clashing window (visibility clash).

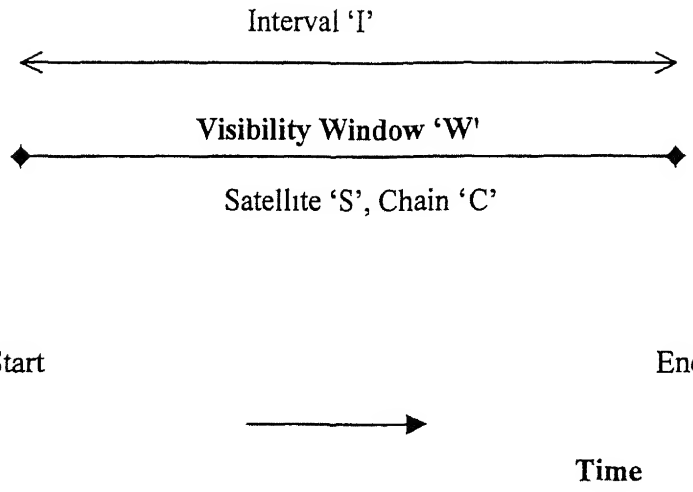
Propagating Visibility Clash Constraint Suppose, W is scheduled to be supported. This implies that the chain C is not available to support any other satellite for time interval I. This implication is imposed on all other visibility windows (say W') on chain C that fall under any of the following cases.

A. 'AOS' of W' $\in$ (-∞, start- Reconfiguration Time] AND 'LOS' of W' $\in$ I
Propagation Of Visibility Clash Constraint Is Done As Convert LOS of this overlapping window W' to 'Start'.

B. 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ [end + Reconfiguration Time, +∞)
Propagation Of Visibility Clash Constraint Is Done As Change AOS of this overlapping window W' to 'End'.

C. 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ I
Propagation Of Visibility Clash Constraint Is Done As: Change status of all operations in this overlapping window W' to 'n'

Figure 5.11: Inter-Station Clash Scenario:



Inter-Station Clash Scenario: 'W' is a visibility window of satellite S at a chain 'C' If at the same time, any part of the visibility window 'W' is visible from a chain other than 'C', then status of window 'W' is acknowledged as a clashing window (inter-station clash)

Propagating Inter-Station Constraint: Suppose, W is scheduled to be supported. This implies that for interval I the satellite S is not available to be supported by any chain other than C This implication is imposed on all other visibility windows (say W') of S on chains other than C that fall under any of the following cases

A. 'AOS' of W' $\in$ (-∞, Start] AND 'LOS' of W' $\in$ I

Propagation Of Inter-Station Constraint Is Done As: Convert LOS of this overlapping window W' to 'Start'

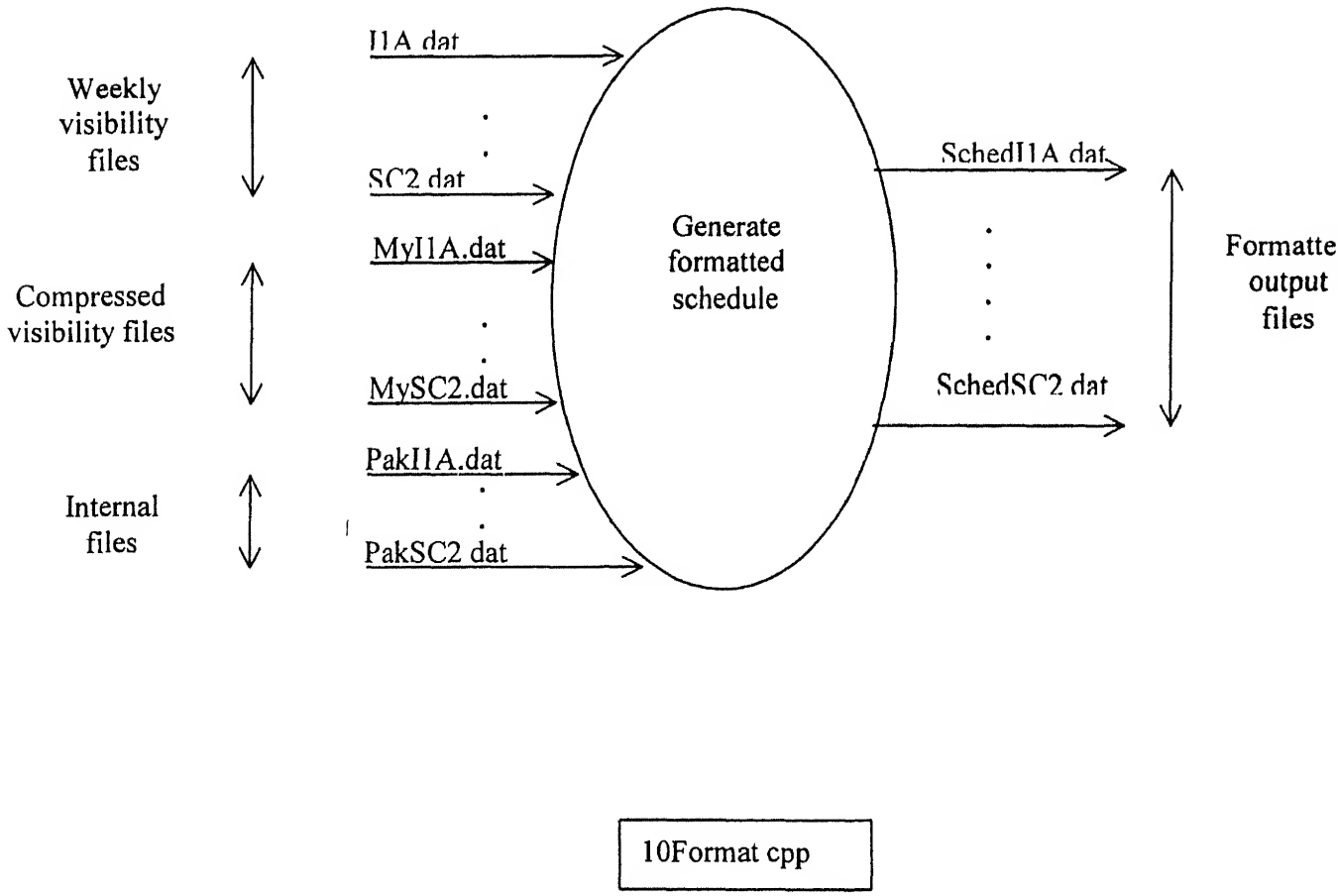
B. 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ [End, + ∞)

Propagation Of Inter-Station Constraint Is Done As: Change AOS of this overlapping window W' to 'End'

C. 'AOS' of W' $\in$ I AND 'LOS' of W' $\in$ I

Propagation Of Inter-Station Constraint Is Done As. Change status of all operations in this overlapping window W' to 'n'.

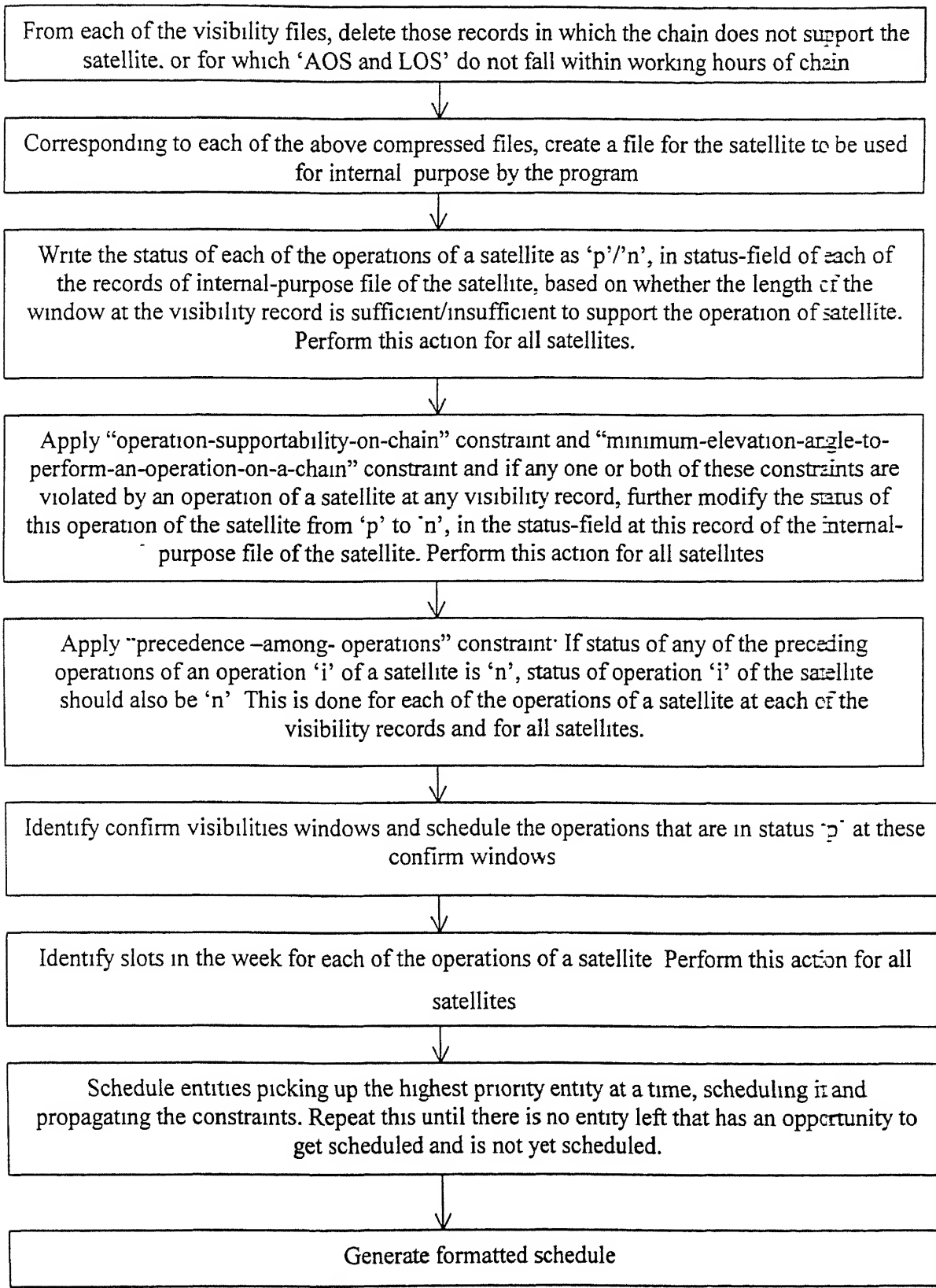
Figure 5.12


Weekly visibility files
Compressed visibility files
Internal files
I1A dat
SC2 dat
MyI1A.dat
MySC2.dat
PakI1A.dat
PakSC2 dat
Generate formatted schedule
SchedI1A dat
SchedSC2 dat
Formatte output files
10Format cpp

5.4 Summary Of Implementation Steps:

Total scheduling process thus passes trough a number of steps. Each of these steps has been implemented through above programs coded in C++. The programs are designed to run on Windows 95 platform. The implementation process extracts information about the names of satellites, names of operations of a satellite, the cyclicity desired of the various operations, minimum duration required to perform an operation, the chains possessed by a TTC station, operations that can be performed on a chain, chain- reconfiguration times, names of satellites under the same controller, controller-preparation-time, etc. from the six input data files. The program takes around 90 minutes on a Pentium III 450 MHz machine, to arrive at the final schedule The steps involved in total process of scheduling are summarized in block diagram 5.13.

Figure 5.13
THE SCHEME FOR SCHEDULING TTC SUPPORT



Our work uses the example of scheduling operations of IRS (Indian Remote Sensing) satellites that are performed by TTC (Telemetry, Tracking and Commanding) stations of ground network The problem is highly complex- the complexity rises exponentially with the number of variables A very large number of variables (satellites, stations, operations, opportunity windows) are involved. Number of constraints is also very high and the priorities of operations change dynamically Further, nonlinear nature of constraints adds to the complexity. To tackle these multiple facets, we have proposed a constraint-based heuristic, discussed in Chapter 3 In this chapter we discuss the results we obtained on implementing this heuristic approach

The metrics of performance we have focused on is "number of entities", an entity being a satellite-operation-slot combination. Our approach attempts to schedule maximum number of these entities, respecting various hard constraints - controller availability, chain availability, precedence relationship, and many more; these have been discussed in Chapter 2

We start with counting number of contending entities, that is, different 'satellite-operation-slot' combinations competing to be scheduled. Table 6.1 on next page, summarizes various *operations* performed on the satellites and Table 6.2 shows number of *entities* corresponding to each of the satellite's operations, to be done at TTC stations in the reference week. In table 6.1, it can be observed that some operations on satellites are to be performed very frequently - once every orbit.

For an operation of a satellite, if the date when this operation is to be performed does not fall within the reference time of one week, there is no slot available, hence the number of entities is zero. Or, if the operation has no defined periodicity (TC operation, discussed in Chapter 2), and is just to be performed as a preceding operation of some other operation, then also number of entities for this operation of a satellite is zero In the eight – satellite scenario, total number of entities present for one week were found to be more than 1300.

In following paragraphs, we show data for number of entities scheduled and number of entities missed, to indicate the effectiveness of the proposed procedure.

Table 6.1: Operations Required by the Different Satellites

SATELLITE		OPERATION NUMBER										
		1	2	3	4	5	6	7	8	9	10	11
IRS-1A	OP NAME	TM	TC	TR	PB	VHF_TC						
	CYCLE	TED:1	-	TED 3	TED 1	OEC						
IRS-1B	OP. NAME	TM	TC	TR	PB	DW	PYS	VHF_TC	DTGLIST	SSP_ON	SSP_OFF	SS
	CYCLE	ENO:3	-	ENO:1	ENO:1	OEC	TED 2	OEC	OEC	TED:1:m	TED:1:n	OEC
IRS-1C	OP NAME	TM	TC	TR	PB	PYS	PYS_RST	CS_RST				
		ENO.3	-	ENO:1	ENO:1	TED 2	OEC	OEC				
IRS-1D	OP. NAME	TM	TC	TR	PB	PYS	PYS_RST	ELP_RST				
	CYCLE	ENO:3	-	ENO:1	ENO:1	TED.2	OEC	OEC				
IRS-P2	OP. NAME	TM	TR									
	CYCLE	TED 2	TED 2									
IRS-P3	OP. NAME	TM	TC	TR	PB	PYS	PYS_RST					
	CYCLE	ENO:3	-	ENO:1	ENO:1	TED 2	OEC					
IRS-P4	OP. NAME	TM	TC	TR	PB	PYS	PYS_RST	PYS_TMR_RST	SPS_PB	SPS_RT	DCS_ES_ON	DCS_ES_OFF
	CYCLE	ENO.3	-	ENO:1	ENO:1	TED 2	OEC	OEC	TED 1.n	TED 1.m	OEC	OEC
SRC-C2	OP. NAME	TM	TC	TR	DW	RPA	GRB					
	CYCLE	ENO:3	-	ENO:1	TED 1	TED 2	TED 1					

OP.=> Operation, ' - ' => no periodicity, TED: n=> n times every day, OEC => once every cycle ,
TED:1:m=> once every day (morning) , TED:1:n=> once every day (night) , ENO: n => once every 'n' orbits

Table 6.2: Number Of Entities

SATELLITE	OPERATION NUMBER											N
	1	2	3	4	5	6	7	8	9	10	11	
IRS-1A	7	0	14	7	0							28
IRS-1B	33	0	84	84	1	14	0	1	7	7	0	231
IRS-1C	33	0	93	93	14	0	1					234
IRS-1D	34	0	97	97	14	1	1					244
IRS-P2	14	14										28
IRS-P3	33	0	98	98	14	0						243
IRS-P4	34	0	91	91	14	0	1	7	7	0	0	245
SRC-2	28	0	70	7	14	7						126
TOTAL ENTITIES												1379

The operation number does not exist for the satellite

N Number of entities

6 1 We defined a priority value function for an entity, in Chapter 3 as . -

$$\text{Priority Value (entity}_I) = (k1 / (\text{number of options in the slot of entity}_I + 2)) + k2 + n^* D \quad \ldots. (1)$$
$$= f1(\text{number of options in the slot of entity}_I) + f2(n, D)$$

As an entity is selected and scheduled, number of options in the slot for other entities gets affected (reduced because of blocking of windows, as discussed in Chapter 3), thus changing the priorities of these remaining entities. First part of the function 'f1' is a decreasing function, its value decreases with the number of options in the slot of entity Values of 'k1' and 'k2' have been chosen arbitrarily as 100 and 150, respectively We do this to ensure that if we assign a lesser value of f1 (say, f1-150) for relatively less important entities, the priority value still remains positive. Why this is done, will be clear in further paragraphs

Number of options in a slot is the number of windows in the slot 'n' represents number of times scheduling of an entity has been missed in immediately preceding slots and 'D' is increase in priority value of an entity, per immediately previous missed slot. Implementing our constraint-based approach, and the using the function mentioned above, we arrived at the number of entities scheduled/missed (metrics of performance we've focused), as shown in tables below (Table 6.2.1 and Table 6 2.2)· -

Table 6.2.1: Number Of Entities Scheduled

SATELLITE	OPERATION NUMBER											N1
	1	2	3	4	5	6	7	8	9	10	11	
IRS-1A	7	0	14	7	0							28
IRS-1B	30	0	64	64	1	14	0	1	7	7	0	188
IRS-1C	33	0	78	79	14	0	1					205
IRS-1D	34	0	74	75	14	1	1					199
IRS-P2	13	13										26
IRS-P3	32	0	82	81	14	0						209
IRS-P4	34	0	76	77	14	0	1	7	7	0	0	216
SRC-2	22	0	37	7	12	7						85
TOTAL ENTITIES SCHEDULED												1156

Table 6.2.2: Number Of Entities Missed

SATELLITE	OPERATION NUMBER											N2
	1	2	3	4	5	6	7	8	9	10	11	
IRS-1A	0	0	0	0	0							0
IRS-1B	3	0	20	20	0	0	0	0	0	0	0	43
IRS-1C	0	0	15	14	0	0	0					29
IRS-1D	0	0	23	22	0	0	0					45
IRS-P2	1	1										2
IRS-P3	1	0	16	17	0	0						34
IRS-P4	0	0	15	14	0	0	0	0	0	0	0	29
SRC-2	6	0	33	0	2	0						41
TOTAL ENTITIES MISSED												223

The operation number does not exist for the satellite

N1 Number of entities scheduled

N2 Number of entities missed

Data tabulated in tables 6 2 1 and 6 2.2 is for D=200. Choosing this value of D ensures that if a competing entity (number of options in the slot of the entity >= 1) got missed in an immediately previous slot, it's priority value becomes highest among those that were not missed in their previous slot. From the tables it can be observed -

- Major fraction of entities got scheduled, and few of the entities could not get allocated at an opportunity window of their respective slots
- Operation numbers 3 and 4 got missed several times. Also, from table 6 1, it is obvious that for a satellite, these operations have highest number of slots, and therefore demand to be scheduled highest number of times in the week, as compared to other operations of the satellite These operations, missed several times, are playback (PB) and *tracking* (TR) operations respectively and cycle of each of these operations is *once every orbit*
- Besides PB and TR operations. some other operations of the satellites are also missed

2) We discussed in Chapter 4 that skipping the operations – PB and TR in some orbits (slots) can be tolerated So, in next step we dilute the priority values of the entities corresponding to PB and TR operations, demanding to get scheduled in every orbit, by using the same priority value function given in equation (1), but with slight modification in constants: -

$$\text{Priority Value (entity}_I\text{)} = (k1 / (\text{number of options in the slot of entity}_I + 2)) + k2 - k + n^* D' \quad \ldots (1)'$$
$$= f1(\text{number of options in the slot of entity}_I) + f2(n, D')$$

Here, D'<D

Introducing 'k' dilutes first part of the priority function of equation (1) – therefore PB or TR operation demanding to be performed 'once every orbit', with some number of opportunities, say 'o', in its slot (orbit) has lower value of priority function (first part) as

compared to that of any other entity with same number of opportunities 'o' in its slot Taking D'<D, (D' is value of increase in priority value of an entity, per immediately previous missed slot for the operations PB and TR of cycle 'once every orbit' and D is value of increase in priority value of an entity, per immediately previous missed slot for the remaining operations) takes care of the fact that more delay can be tolerated for PB and TR operations demanding to be performed most frequently - once every orbit Taking values of k and D' as 150 and 35 respectively, we arrived at the following number of entities scheduled/missed (Table 6.3.1 and Table 6.3.2)

Comparing these tables with the previous (Table 6.2 1 and Table 6 3.2) ones, we find that entities for some operations, other than operation numbers 3 and 4 (i e PB and TR with cycle 'once every orbit) that were missed have got scheduled – entities for operation number 1 and 2 of satellite IRS-IP2 and for operation number 5 for satellite SRC-C2. This occurs because the opportunities for these entities were grabbed away by PB and TR operations, cycles of which is 'once every orbit' Using the information, that missing of PB and TR operations with cycles specified as 'once every orbit', can be tolerated, we adjusted the parameters.

Table 6.3.1: Number Of Entities Scheduled

SATELLITE	OPERATION NUMBER											N1
	1	2	3	4	5	6	7	8	9	10	11	
IRS-1A	7	0	14	7	0							28
IRS-1B	30	0	60	58	1	14	0	1	7	7	0	178
IRS-1C	33	0	76	76	14	0	1					200
IRS-1D	34	0	71	73	14	1	1					194
IRS-P2	14	14										28
IRS-P3	33	0	81	81	14	0						209
IRS-P4	34	0	57	77	14	0	1	7	7	0	0	197
SRC-2	22	0	37	7	13	7						86
TOTAL ENTITIES SCHEDULED												1120

Table 6.3.2: Number Of Entities Missed

SATELLITE	OPERATION NUMBER											N2
	1	2	3	4	5	6	7	8	9	10	11	
IRS-1A	0	0	0	0	0							0
IRS-1B	3	0	24	26	0	0	0	0	0	0	0	53
IRS-1C	0	0	17	17	0	0	0					34
IRS-1D	0	0	26	24	0	0	0					50
IRS-P2	0	0										0
IRS-P3	0	0	17	17	0	0						34
IRS-P4	0	0	34	14	0	0	0	0	0	0	0	48
SRC-2	6	0	33	0	1	0						40
TOTAL ENTITIES MISSED												259

The operation number does not exist for the satellite

N1 Number of entities scheduled

N2 Number of entities missed

Entities for operation numbers 1 and 5 could not be scheduled even after adjustment because no opportunities were found in slots right from the initial stage – that is, after forcing following constraints and maintaining the status of operations at each of the visibility windows -

- "Satellite supportability on chain" constraint
- "Station working hours " constraint
- "Minimum duration of operation " constraint
- "Operation supportability on chain" constraint
- "Minimum elevation to perform an operation on a chain "constraint
- "Precedence" constraint

Thus, we see that a tradeoff exists when scheduling entities of various satellites and for all operations of satellites – scheduling one operation grabs away chance of scheduling other operations. These tradeoffs can be viewed from figures 6.1 to 6.8. In these figures series 2 is after parameter adjustment . From these figures it is obvious that

Figure 6.1: IRS-1A

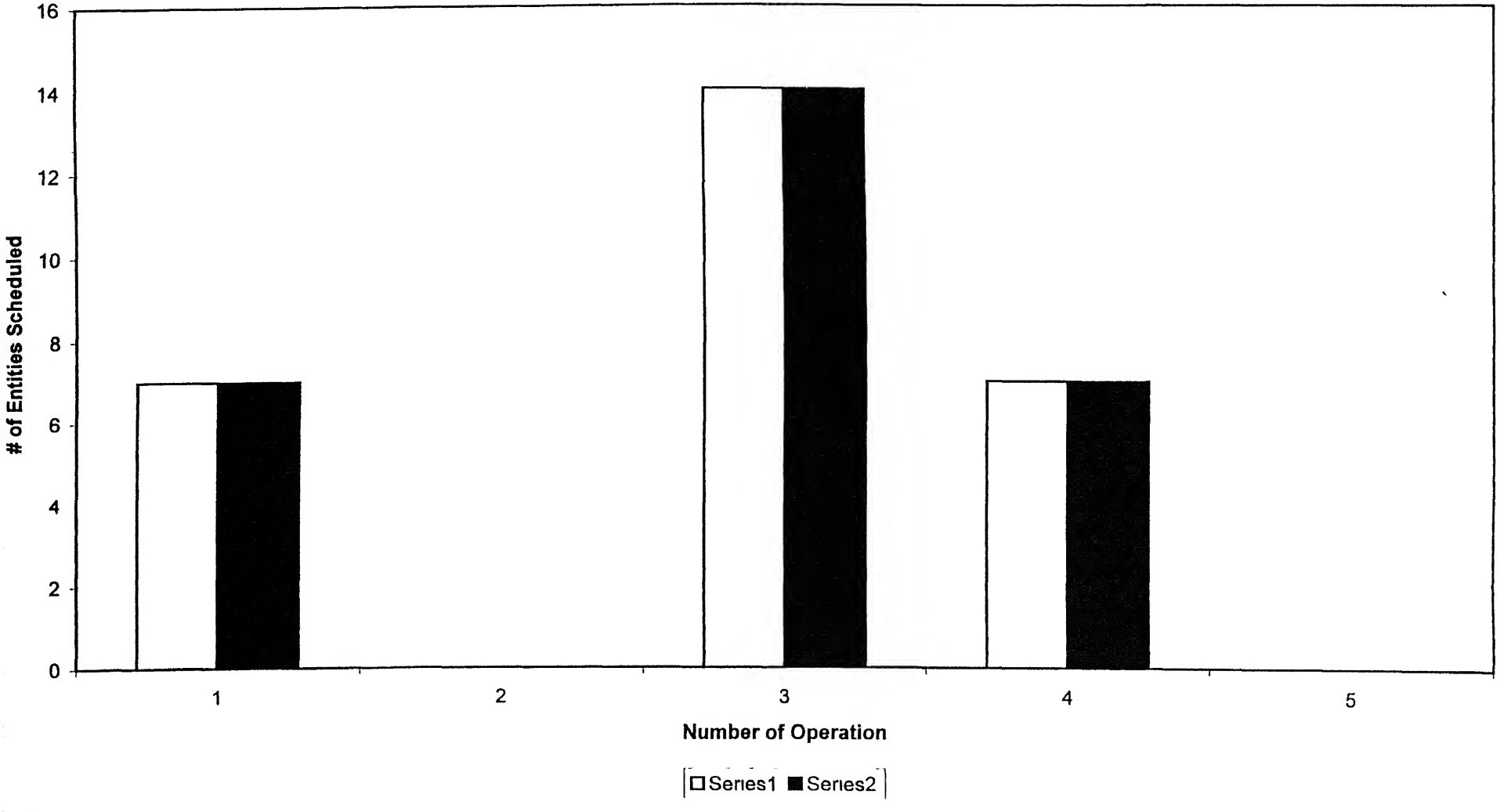


Figure 6.2: IRS-1B

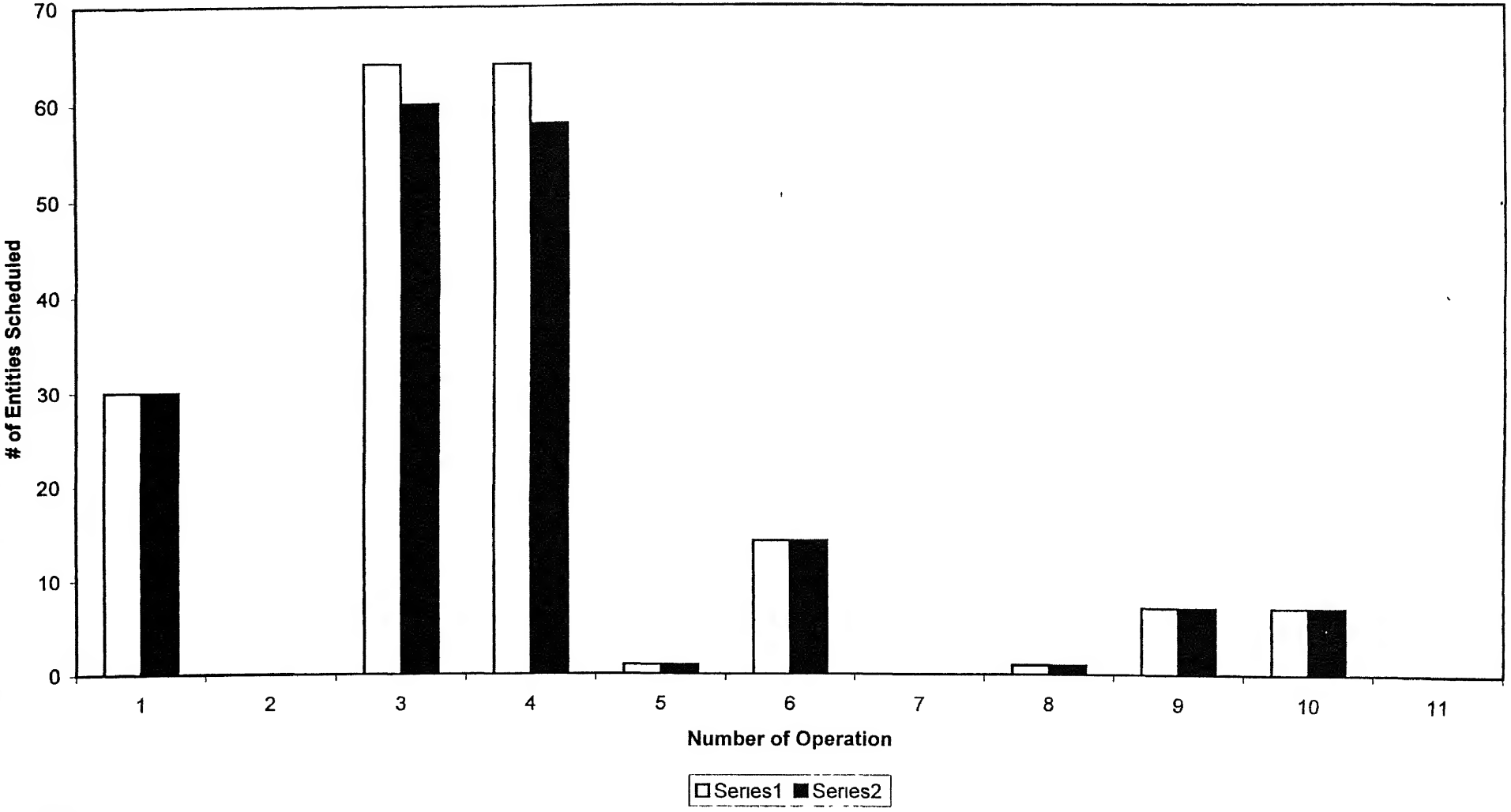


Figure 6.3: IRS-1C

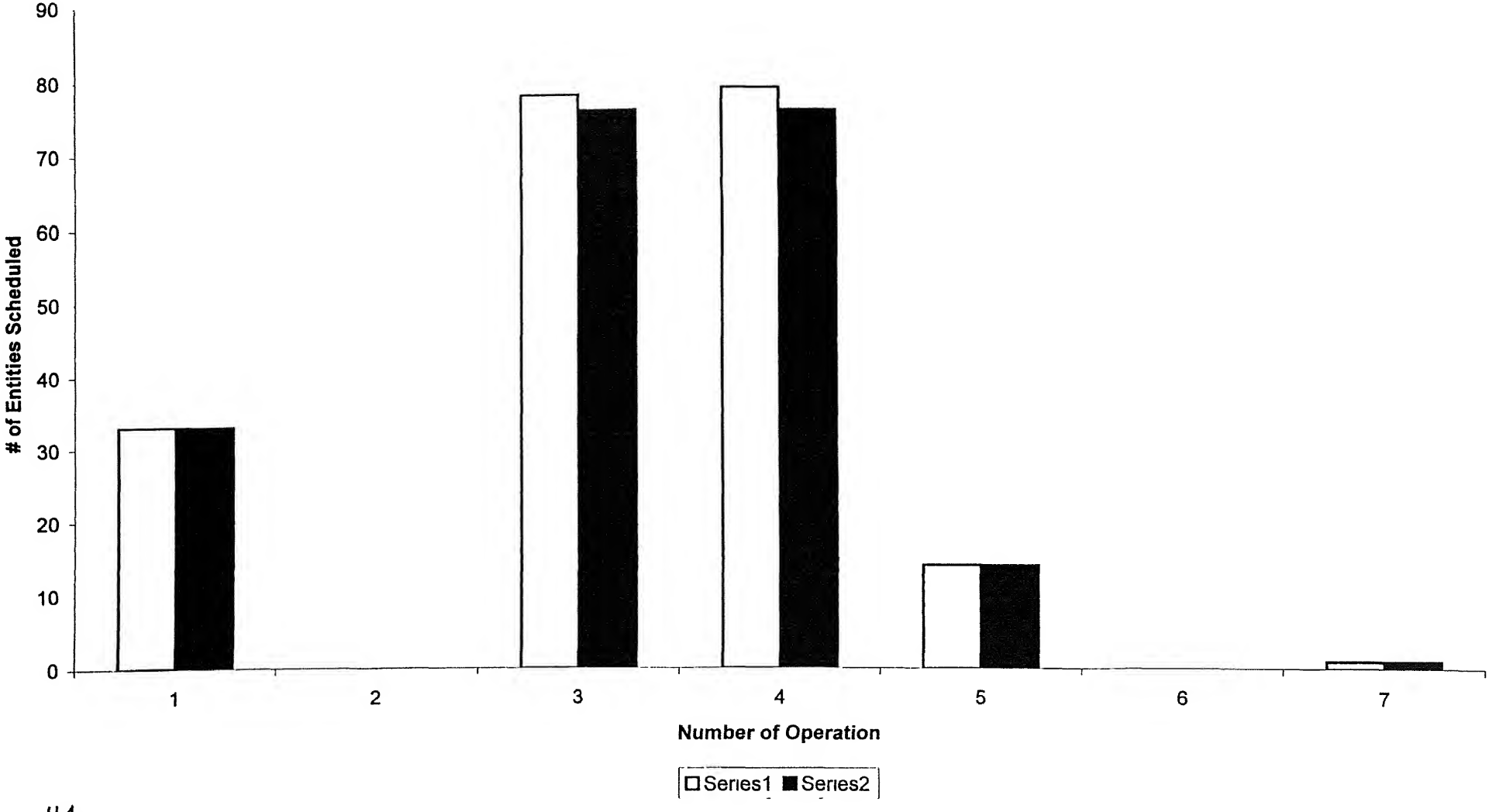


Figure 6.4: IRS-1D

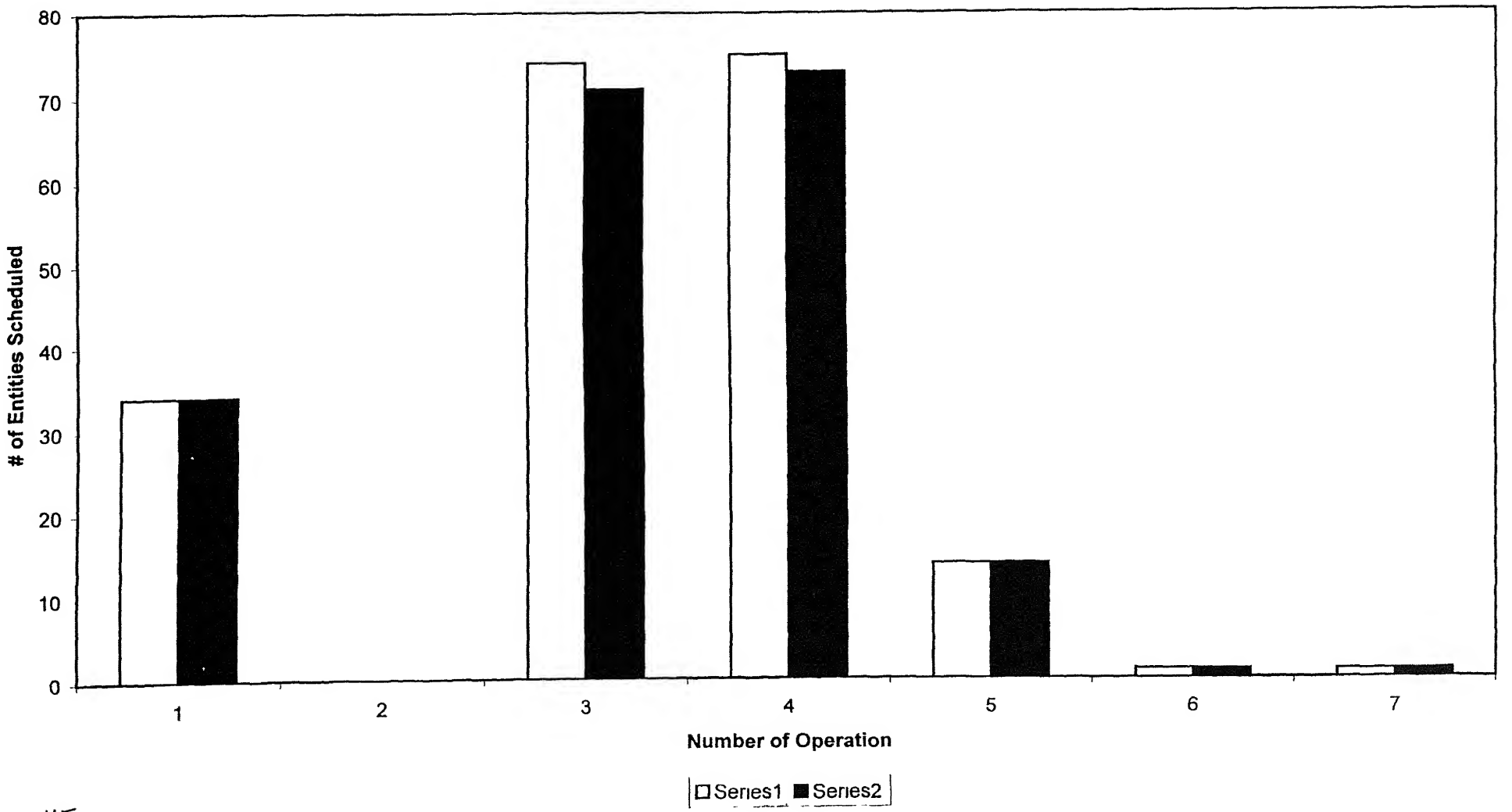


Figure 6.5: IRS-P2

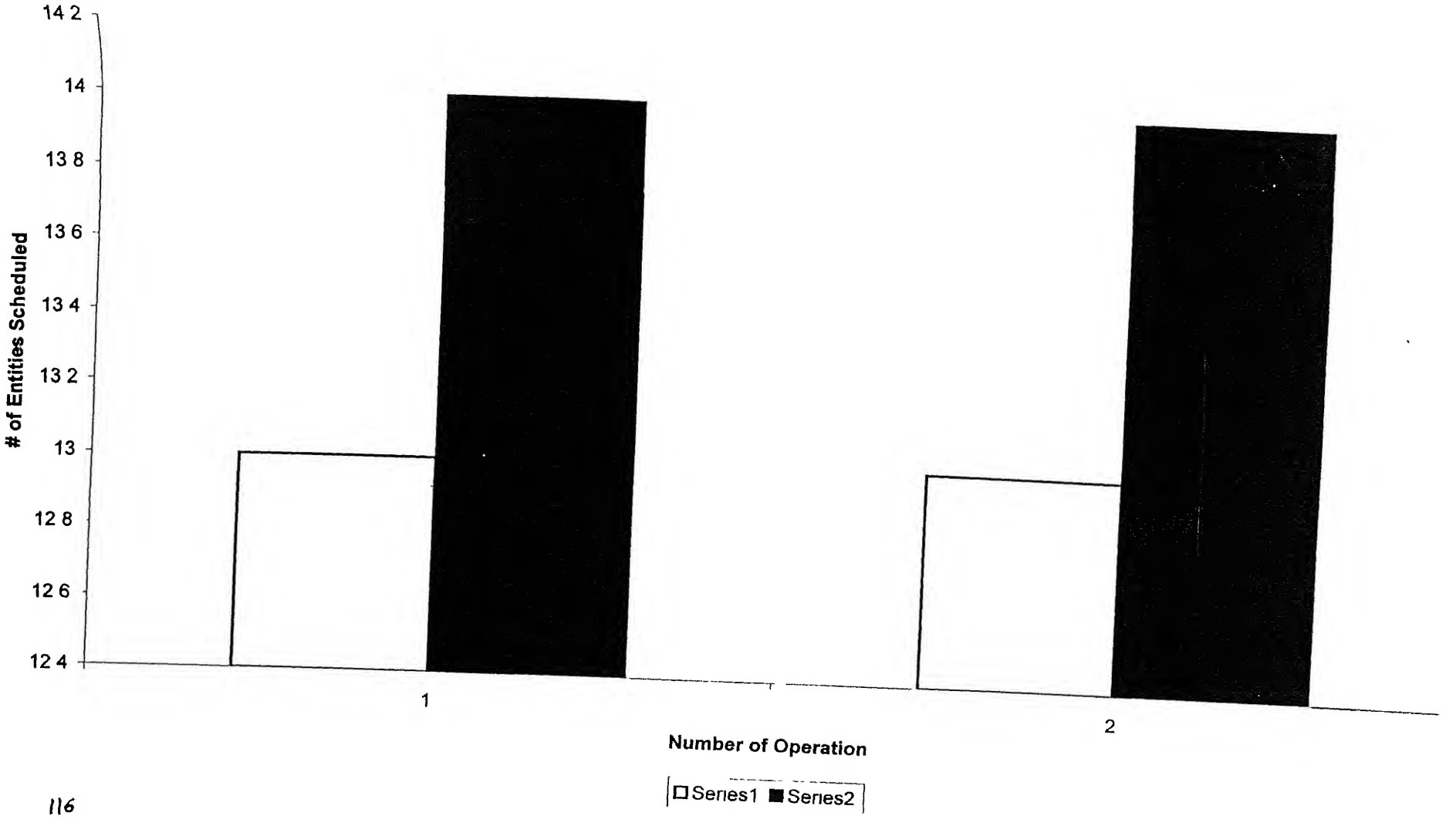


Figure 6.6: IRS-P3

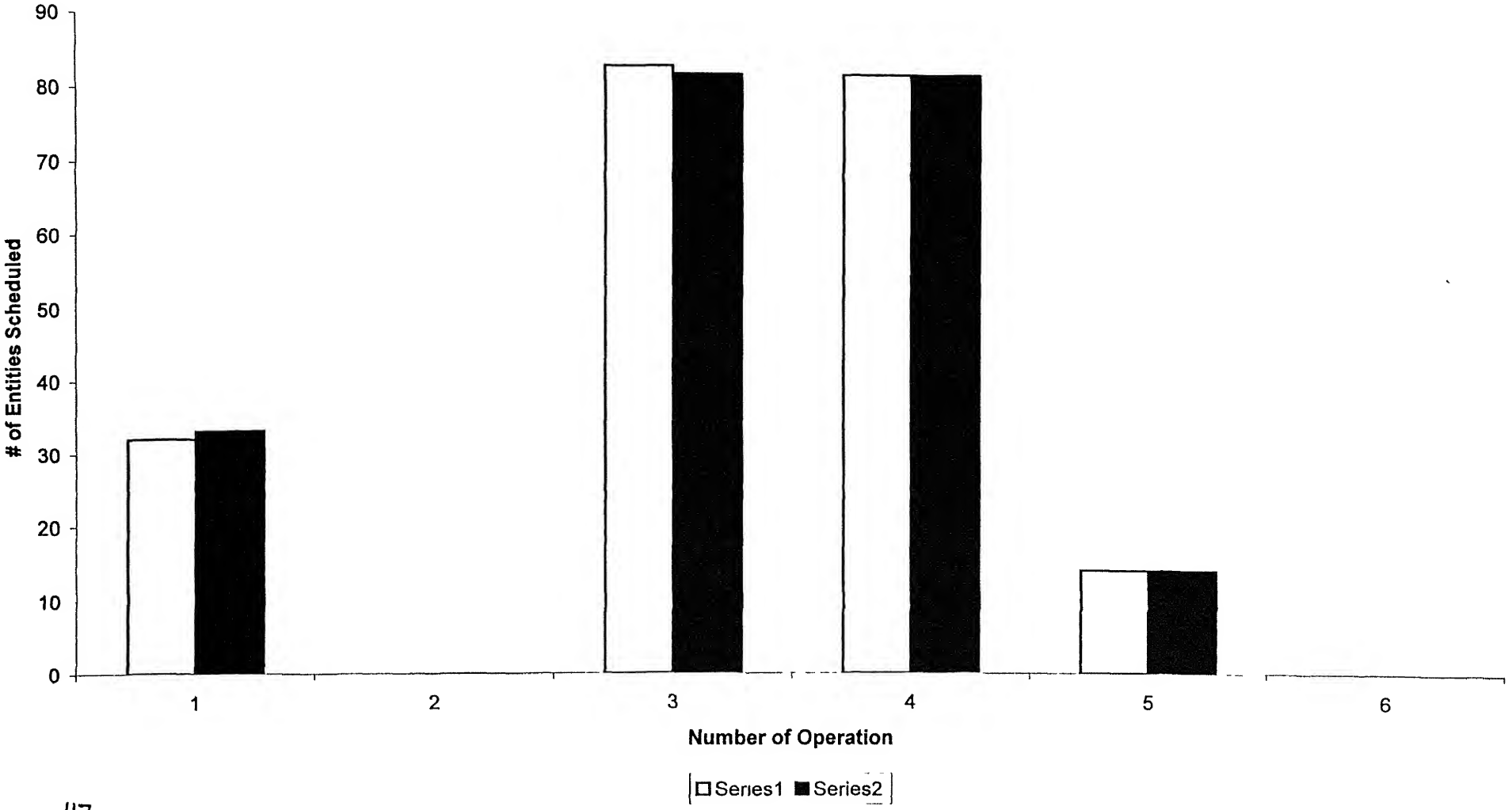


Figure 6.7: IRS-P4

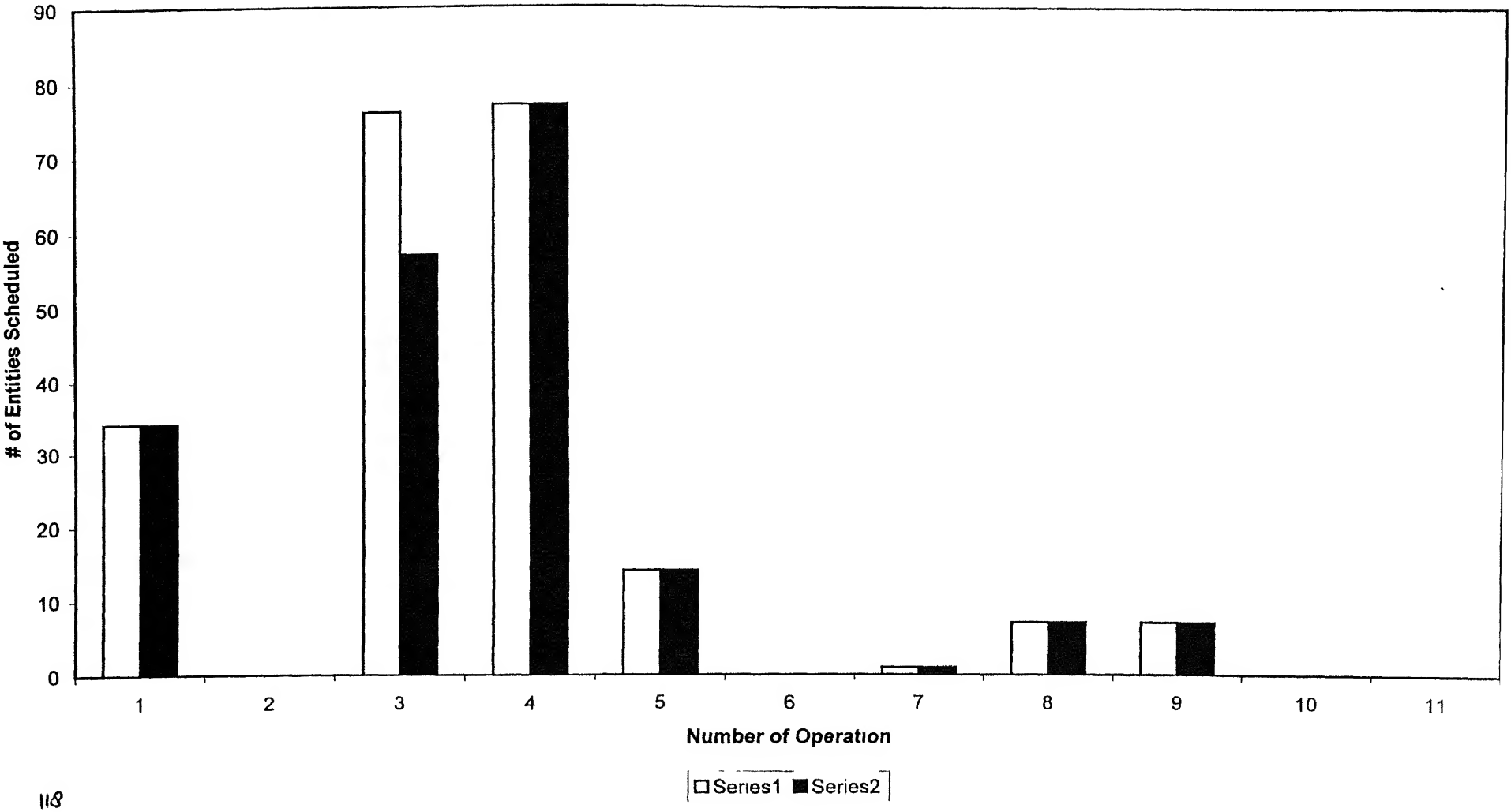
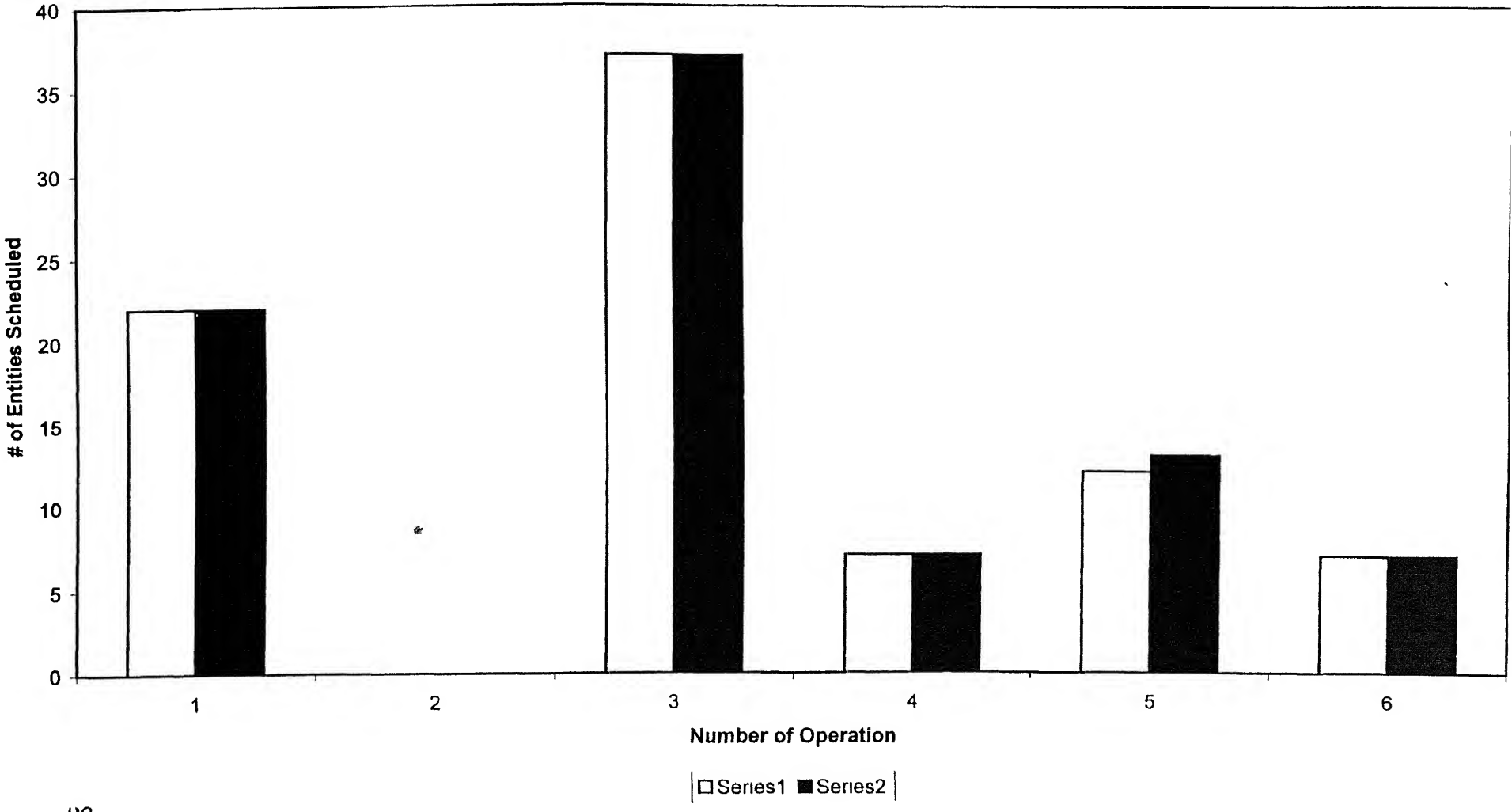


Figure 6.8: SRC-2



on parameter adjustment, the operation numbers 3 and 4 (PB and TR: high frequency operations, skipping of which in some slots can be tolerated) get scheduled same or less number of times on the satellites On the other hand, operation numbers 1 and 2 (TM and TR) of IRS-P2 and operation number 5 of SRC-C2, that are more important operations as compared to PB and TR get scheduled more number of times (Figure 6 4 and 6 8)

These results suggest that adjustment of parameters affects the schedule. If this adjustment is done carelessly, the dispatcher may produce poor schedules –pushing out important operations and scheduling the lesser important ones.

In this chapter we have quantitatively evaluated the effectiveness of the constraint-guided search method, and its sensitivity to the decision-guiding parameters

In this thesis, we reviewed the mission operations of LEO satellites and the various constraints specific to challenge of scheduling of these operations in the most acceptable manner possible while meeting resource availability constraints, temporal constraints, logical constraints and other constraints. Each of these operations of satellites is required to be performed in desired periods of time and the priorities of these operations may change dynamically The number of constraints is large (the problem is over -constrained) so that it is difficult to arrive at a feasible solution within a reasonable time The non-linear nature of some of the constraints imparts more complexity to the problem, thus making the problem more difficult to solve.

To tackle the complexities posed by the multitudes of constraints, the nature of these constraints, and dynamically changing priorities, we applied here a constraint-directed search heuristic We constructed a knowledge representation capturing constraints (controller preparation time, chain reconfiguration time, etc) and other factors (satellite and task priority) that have impact on the scheduling process. We have developed a search procedure capable of handling the dynamic priorities of operations and constraint knowledge. The procedure utilises constraint propagation tools to prune the search space and is thus more flexible and accommodative as compared with traditional algorithmic techniques of optimization. We implemented the method for a eight-satellite scenario and arrived at a feasible solution that to the user is "good enough" It successfully allocated windows for various telemetry, tracking and commanding operations of satellites according to their relative importance and at their specified periods of time – as indicated by the performance statistics determined at the end Over 85% of the tasks could be scheduled for the example studied

The current research has some limitations that suggest valuable directions for future research: -

1) Presently, we have considered the preference of placement within a window as uniform, whereas the scientific worth of taking an observation within a window may be higher towards the middle part of the window (satellite is at highest elevation at middle part of the window and therefore the observations and communications are of better quality in this part).

2) We have considered the planning horizon for scheduling the operations as one week. This period should be flexible

3) Some other features of a good schedule such as, the uniformity of work- load on ground stations has also not been considered. These features may be incorporated with appropriately constructed objective functions. However, the higher the number of such options considered, the higher becomes the complexity of the scheduling problem In such case, meta-heuristic search may be incorporated

We have discovered computational limitations to tackling the problem as done here. The scenario for which we implemented the heuristic contains only eight satellites with each satellite tied to a set of operations to be performed at specified periods The time taken by the code developed to arrive at the final schedule is around 90 minutes, which is clearly very large

Constraint based approach is already noted to be a very powerful tool to solve large and complex problems. Multi-satellite-scheduling is one such example. Much work has been done in this field and yet much more is still left to be explored, for, even though the method does not guarantee optimality of the schedule, it is capable of producing reasonably *good* schedules. Particularly, further strategies should be explored to reduce the search effort, such as intelligent and superior methods for constraint representation and constraint propagation.

References

[1] K Kasturirangan, Developments in Indian Space Programme, *Technorama* (IE India), August 2000, 5-9

[2] J C Agnese, N Bataille, D Blumstein, E Bensana and G Verfaillie, Exact and Approximate Methods for the Daily Management of an Earth Observation Satellite, *Proc 5th Workhsop ESTEC Artificial Intelligence and Knowledge Based Systems for Space*, Noordwijk (NL), 1995, 10-12 October

[3] J C Agnese and Pascal Brousse, 1998 Scheduling Techniques for a Constellation Visibilities, Spaceflight Dynamics 1998, *Proc AAS/GSFC International Symposium*, Greenbelt, MD, 1998, May

[4] K R Baker, *Elements of Sequencing and Scheduling*, Dartmouth College, 1993

[5] SPIKE AI Scheduling for Hubble Space Telescope, www stsci edu/apsb/doc

[6] N Groleau, LKiser, F Girouard, A fully implemented semi-automated ground-control system for the Terriers satellite, NASA Ames Research Center, 1992.

[7] GREAS, The Satellite Resource Management Tool, www psrw com/GREAS/grweb2 html

[8] J C Pemberton and F Galiber III, A constraint-based Approach to Satellite Scheduling, Pacific Searra Research, 1998

[9] www monet astro uiuc edu/adass98/proceedings/kleineresc/

[10] W J Wolfe and S E Sorensen, Three Scheduling Algorithms Applied to the Earth Observing Systems Domain, *Management Science*, January 2000.

[11] Sanjay Kumar, *LEO Satellite Operations Scheduling· An Approach using Genetic Algorithms*, M Tech Thesis, Indian Institute of Technology, Kanpur, 2000.

[12] M Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, NJ, 1995

[13] T P Bagchi and Kalyanmoy Deb, Calibration of GA Parameters: The Design of Experiments Approach, *Computer Science and Informatics*, 26, 3, 1996

[14] T P Bagchi, *Multiobjective Scheduling by Genetic Algorithms*, Kluwer Academic, MA, 1999.

[15] J D Rao, P Soma and G S Padmashree, Multi-Satellite Scheduling System for LEO Satellite Operations, 2b002, *Proceedings*, Space Ops 98, 1998.

[17] Monte Zweben, Mark S Fox (1994), "Intelligent Scheduling", Morgan Kauffmann Publishers, San Francisco, California

[18] Web Reference 1 **www.cpc.unc.edu/services/spatial/remsens.htm**

[19] http //budhı uow edu au/staff/aditya/dsl/dsl-cp html

[20] ILOG Optimization Suite: White Paper, April 1998

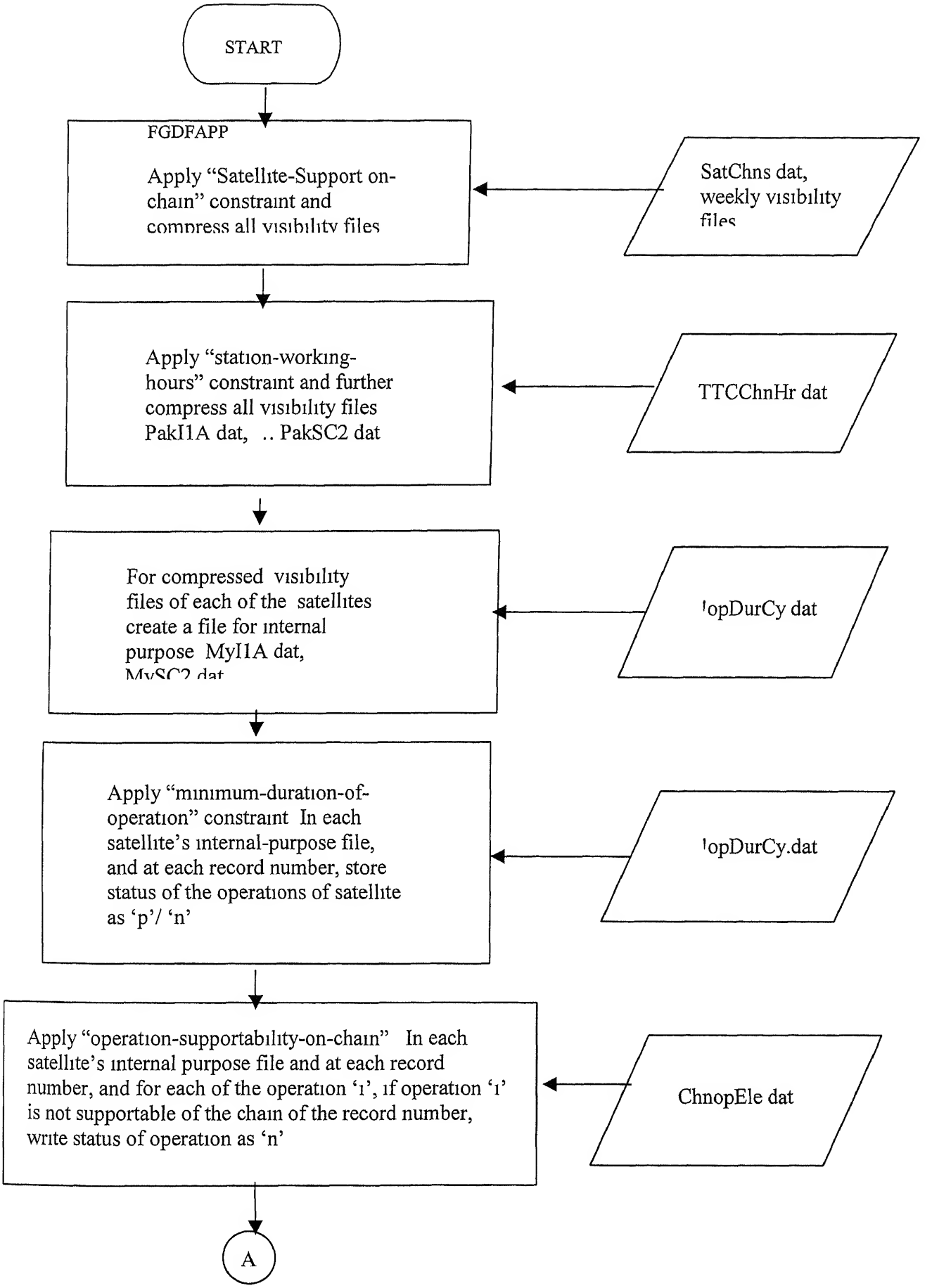
[21] Web Reference 2 **http://www.ilog.fr/products/solver/**

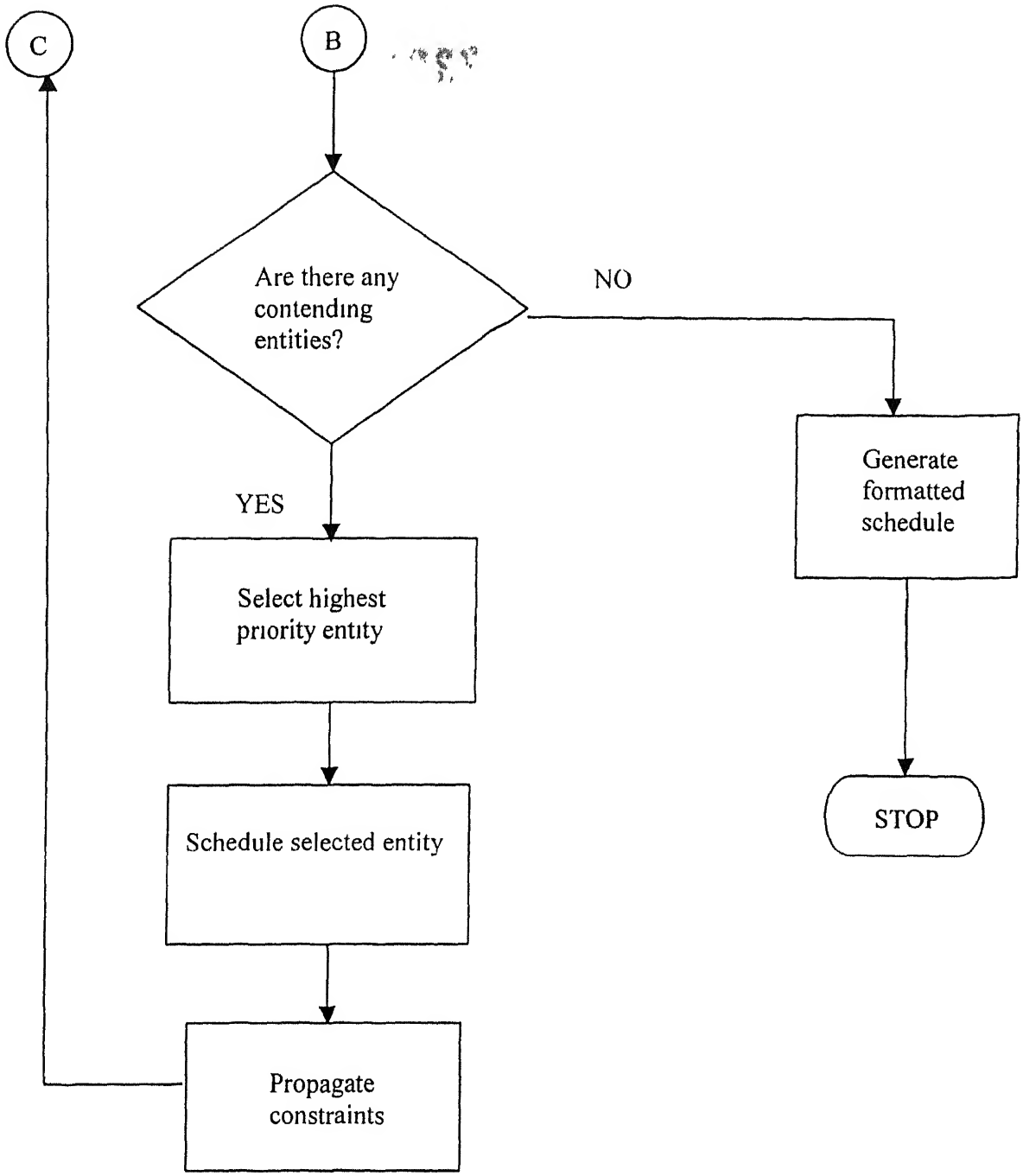
[22] Web Reference 3. **www.braxtontech.com/VisualScheduler.html**

[23] Web Reference 4. **http://budhi.uow.edu.au/staff/aditya/dsl/dsl-cp.html**

[24] Web Reference 5 www ıcparc ıc ac uk/eclipse/reports/handbook/node15.html

FLOW CHART


START
FGDFAPP
Apply "Satellite-Support on-chain" constraint and compress all visibility files
SatChns dat, weekly visibility files
Apply "station-working-hours" constraint and further compress all visibility files PakI1A dat, .. PakSC2 dat
TTCChnHr dat
For compressed visibility files of each of the satellites create a file for internal purpose MyI1A dat, MySC2 dat
!opDurCy dat
Apply "minimum-duration-of-operation" constraint In each satellite's internal-purpose file, and at each record number, store status of the operations of satellite as 'p'/ 'n'
!opDurCy.dat
Apply "operation-supportability-on-chain" In each satellite's internal purpose file and at each record number, and for each of the operation '1', if operation '1' is not supportable of the chain of the record number, write status of operation as 'n'
ChnopEle dat
A


C
B
Are there any contending entities?
NO
YES
Generate formatted schedule
Select highest priority entity
Schedule selected entity
STOP
Propagate constraints